BRL-CR-638

(2)

CONTRACTOR REPORT BRL-CR-638

BRL

TANKWARS FOR THE PARAMETRIC CONSIDERATION OF SYSTEM CONCEPTS

HARRY L. REED, JR.
BATTELLE COLUMBUS
COLUMBUS, OHIO

AUGUST 1990

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

AD - A227 284

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
August 1990

3. REPORT TYPE AND DATES COVERED
Final 10/87 to 02/90

4. TITLE AND SUBTITLE
TANKWARS for the Parametric Consideration of System Concepts

5. FUNDING NUMBERS
PE: AH80
CR: DAAL03-86-D-0001

6. AUTHOR(S)
Harry L. Reed, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Battelle Columbus
505 King Avenue
Columbus, OH 43201

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
BRL-CR-638

11. SUPPLEMENTARY NOTES
Task was performed under a Scientific Services Agreement issued by Battelle, Research Triangle Park Office, 200 Park Drive, P.O. Box 12297, Research Triangle Park, NC 27709

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report discusses aspects of TANKWARS relevant to the future armament concepts for armored vehicles. TANKWARS was picked as a good tool for starting this effort. TANKWARS is a Monte Carlo computer simulation of engagements between two homogeneous mechanized forces. Recommendations are made of possible modifications to TANKWARS or the need for a new model. This report should be useful as an adjunct to documentation on TANKWARS or on GROUNDWARS.

14. SUBJECT TERMS
TANKWARS, GROUNDWAR, SYSTEM ANALYSIS, TANKS, TANK GUNS

15. NUMBER OF PAGES
120

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
UL

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

Intentionally left blank.

TANKWARS for the Parametric Consideration of System Concepts

by

Harry L. Reed, Jr.

INTRODUCTION

The core of the effort was to develop and exercise a methodology for the analysis of future armament concepts for armored vehicles to allow the assessment of the application of new technology to those weapon systems. TANKWARS was picked as a good tool for starting this effort. Thus TANKWARS as adapted and used to carry out numerous parametric studies of tank armament concepts. The data from these studies were furnished as developed to Dr. Howe.

The data devcloped are of little use to anyone who is not aware of the classified issues involved, and in many instances the data represent stages in the thinking about what the methodology should do and what issues should be given attention for further analysis; thus Dr. Howe decided that no useful purpose would be served in including those data in this report. Of course, the data are being used by Dr. Howe to develop his recommendations for the Army's Technical Base Program in armaments.

OBJECTIVE

This report then will discuss aspects of TANKWARS relevant to the above work, changes that were made in the TANKWARS program, ideas for the future application of TANKWARS (or possibly its successor) in further parametric analyses, and some thoughts on the development of more advanced tools for modeling. This report should be specially useful as an aajunct to documentation on TANKWARS or on GROUNDWARS.

1. TANKWARS - A Computer Simulation

TANKWARS is a Monte Carlo computer simulation of engagements between two homogeneous mechanized forces. The model simulates individual weapon systems,and the engagements include search, detection, selection, firing, impact, functional destruction, disengagement, and reengagement. Nominally it can handle up to 20 armored vehicles on each side. The computer program was written by Mr. Fred Bunn of the Ballistic Research Laboratory (BRL), Aberdeen Proving Ground, Maryland (reference 1).

GROUNDWARS is an outgrowth of the TANKWARS model and was developed by the Army Material Systems Analysis Activity (AMSAA). Some of the features of GROUNDWARS are addressed in Section 3. At the time we first discussed GROUNDWARS with AMSAA, version 3.97 was running, version 4.0 was being worked on, and a new concept u:ing the "C" language was in the early stages of development. We have had some very useful discussions with AMSAA people (specially Barry Burns) which we have found most useful and which we hope were of some use to AMSAA.

We felt that TANKWARS provides a reasonable m on n model that includes a level of detail about the systems under consideration, that it could allow sensible engineering conclusions, and that it represents a tactically meaningful environment for giving military significance to any such conclusions.

An important consideration was running time since the parametric analyses require the running a large numbers of cases. TANKWARS appeared to offer a reasonable balance between system detail and tactical context with practical run times. The significance of running time became even more obvious when the computer originally picked for use (a Gould 9000) proved to be too slow, and the work was moved to a Cray X/MP-48. The latest version 4.0 of GROUNDWARS appears interesting in this regard as a considerably speeded up and improved successor to TANKWARS. Initially, GROUNDWARS 3.97 appeared to be more complicated than necessary, and it was too large to fit on the Gould computer (or so the compiler kept saying). Of course, the use of the larger Cray and the recent availability of the faster version of GROUNDWARS are reason for reopening that decision.

Other advantages of TANKWARS were felt to be:

* It is well enough written to allow change - an important consideration since new weapon concepts often require changes in employment for full realization of their potential. This is specially the case in modern fire control concepts which offer concurrent advantages in target acquisition and in command and control.

* TANKWARS is in use in the analysis community and is the basis for AMSAA's GROUNDWARS. Transition to GROUNDWARS 4.0 should be relatively easy if that proves to be desirable. In fact, many of the subroutines are the same.

* Mr. Bunn, the developer of TANKWARS, was easily accessible for numerous discussions on nuances of the program and changes.

One feature of TANKWARS that was not utilized in the effort described in this report was that of modeling sequential engagements and looking at resupply on one side. AMSAA also did not include this feature when they created GROUNDWARS.

As was mentioned before, the general acceptance of TANKWARS is testimony to its usefulness. But there are always needs for more. For the purposes at hand the shortcomings of TANKWARS include:

Only one type of armored vehicle (including one type of weapon and one type of ammunition) is allowed on each side. Thus a mix of weapons that might have application at different ranges in the engagement cannot be played. Neither can different systems in a cooperative arrangement such as tanks advancing with other vehicles in protective overwatch with guided missile systems.

The available scenarios only include an advance by one side or a meeting engagement in which both sides are stationary throughout the engagement. We see the need for the ability to conduct engagements between two moving (and occasionally stopping) forces.

Likewise there is a need for simulating more complicated overwatch tactics such as leapfrogging. As will be noted further on, a simple overwatch tactic has been added in which the overwatching force never moves and has the same weapon as the advancing force.

The decision process for engaging, disengaging, and reengaging also needs to be reworked. For example, the ability might be added to disengage from a low priority target if a higher priority target were to become available. At present the disengagement criteria are uniformly applied based on time, or based on rounds fired, or based on damage to the target, but are not related to the relative importance of the target. The importance of the target comes into play only in the initial selection. Faking disengagement by firing only a small/fixed number of rounds and using the subsequent target selection process to represent the search for a more important target is not an adequate surrogate. The decision to reengage the old target would not be the same as the simple decision to continue the firing sequence since the time to fire the "next" round would be the time to fire a "first" round rather than the shorter time to fire a subsequent round. When runs were made in which target information was shared among the tanks in a force, the result was often negative. This indicates that the additional information was not being used properly by the decision process associated with engagement and disengagement.

Even when done on the X/MP-48, the calculations for vast parametric arrays are somewhat slow. Most of the computing time is spent in the search and related target acquisition process which is repeated every second. The version 4.0 of GROUNDWARS treats this differently and achieves a significant speed increase.

2. Changes to TANKWARS

In talking about the changes made to TANKWARS during this effort, we often refer to the FORTRAN source program listed in Appendix A. That listing is only given to allow one to get an idea of what the program does and more importantly to show the changes that were made. Anyone who is interested in acquiring the code for use should contact Mr. Bunn at BRL for documentation.

The program as shown includes a variety of changes by the author that are often rather crude patches. No uniform attempt has been made to clean up the modified code, clear out any parts that have been rendered useless by the changes, rework the code to make the additions more efficient, and blend the changes into the fabric of the code more smoothly. Since there appears to be good reasons to move to a more advanced program, there would be no good purpose served by such an effort. Also, a very large portion of the run time is spent in the process of searching for targets and updating the positions of the vehicles; so any small inefficiencies introduced into most of the code are not a matter of great significance.

The UNIX environment was used for the minicomputer operation and UNICOS for the operation on the Cray computer.

As cases were run, we found hiccups, bugs, and needs for changes in the program. We also developed a sense of what we should be looking at and looking for.

The parametrics considered include accuracy, lethality, acquisition (regular and pinpoint), rates of fire, time to lay the weapon, decoys, overwatch, decision algorithms, and fire control options. Questions have proliferated more rapidly than answers, and that trend will probably continue.

TANKWARS uses a number of input files. The game file and the miscellaneous file are discussed in considerable detail in reference 1:

* the game file sets up the numbers of players, the ranges of engagement, the types of scenarios (e.g. who is defender and who is aatacker), and other data including the names of files that define the characteristics of the weapon systems being played. For the version of TANKWARS discussed in this report there are two files for each of the two armored vehicles, a misc file and a vul file.

* the misc file for a side gives a variety of data about the vehicle on that side such as its dimensions, speed, ability to detect targets, rate of fire, etc.

* the vul file for a side gives the probabilities of hit and of various types of kill given a shot for the weapon used by that side against the vehicle on the other side. As noted below, the original version of TANKWARS used two files here, one for accuracy and one for conditional probability of kill given a hit.

2.1 Changes for Running

For the convenience of running numerous parametric cases, a shell file approach was created for overall management of the calculation. A few game files were created each of which used the same names for the other input files - namely blue.misc, blue.vul, red.misc, and red.vul.

Then a UNIX shell file (called runtw for example) was created such as

```
cp $2 blue.misc
cp $3 blue.vul
cp $4 red.misc
cp $5 red.vul
```

```
echo Defender: $2";" $3 > $6
echo Attacker: $4";" $5 >> $6
../source/twx1 < $1 >> $6
```

This shell program was called by a command line such as:

```
runtw file1 file2 file3 file4 file5 file6
```

This command line produces what we will call a case in this report. A case is a series of runs each with the number of Monte Carlo replications specified in file1 (the game file) and each with one of the opening ranges specified in file1. The misc file, file2, describes the blue force vehicles. The blue vul file, file 3, contains data on the ability of the blue vehicles to kill the red vehicles. The misc file, file4, is for red. The vul file, file5 is for red kills of blue. The file, file6, is that into which the output data are to be stored.

Finally a number of cases can be run by setting up a shell file containing a number of these command lines.

As a matter of convenience we used the blue force as the defending force and the red force as the attacking force rather than using these terms as indicators of political affiliation.

TANKWARS prints a considerable amount of detailed data about each run made at each opening range. Except for occasional diagnostics, these data were more than were needed, and vast outputs were being produced for the large number of runs. Therefore the indicated change in the subroutine NXWAVE was introduced and used to produce one line of data for each opening range in a case. As that change is shown, that output line contains:

```
      R   Ndef   Ndefdec   Natt   Noverw   Exch   Defrds   Attrds
```

where
R   = the opening range
Ndef   = the average number of defenders (not decoys) killed
Ndefdec = the average number of defender decoys killed
Natt   = the average number of advancing attackers killed
Noverw = the average number of overwatching attackers killed
Exch   = the exchange ratio = Ndef / (Natt Noverw)
Defrds = the average number of rounds fired by a defender vehicle
Attrds = the average number of rounds fired by a defender vehicle

Other formats could be easily tailored as needed.

2.2 Modifications for Decoys and Overwatch

We were concerned with decoys that were stationary and flashing (simulating firing). We also included the ability to have tanks on the side of the attacking force that remained at the opening range in hull defilade; these were called overwatching tanks.

The changes needed for decoys and overwatching tanks are intertwined in the code and are contained in the subroutines CANGO, DEATHS, DEPLO2, FINISH, INIT2, INPUT, and SEARCH. Decoys were already in TANKWARS, but the changes were needed to remove them from win statistics (changed to say that if all the real tanks are killed on a side, that side loses even if the decoys are alive) and to count only real tanks in the exchange ratios.

The changes in INIT2 include ordering the real and decoy tanks so that if the attacking force (called red in the program) has any decoys, the decoys will be in the overwatching role as long as there are no more decoys than there are overwatching tanks. Since we didn't have a good idea on how to treat moving decoys, we only considered stationary ones. The stationary decoys are handled much the way that tanks are handled. They acquire and "shoot" at tanks but no rounds fly to the target, and they are acquired and killed as are tanks.

The change in SEARCH allows the defender's acquisition ability to account for both hull defilade stationary overwatchers and fully exposed moving attackers. We further decided that an attack would be stopped (and thus the attacking side would lose that case if all the advancing tanks were killed regardless of whether the overwatching tanks were killed or not. A more realistic model would include the possibility for the advancing tanks and the overwatching tanks to trade places (leapfrogging).

2.3 Change to Ensure Loading of "First" Rounds

This became a problem when some of the time constants associated with laying the gun became less than the time needed to load a round. As it was written, TANKWARS accounted for the time needed to reload the gun assubsequent rounds were fired at a target. However, no such loading time was "required" from the time the last round was fired at one target until the next round was fired at another target. This was not a problem for the very first round since it was assumed that the tank would go into battle with a round in the tube. Changes were made in the subroutines ENGAGE, HALT, and INIT2. The change in INIT2 was to put in a bogus value for tfire2 which was tmin seconds before the battle would begin. The variable tfire2 is the time that the tank last fired and tmin is the time to reload. Thus the tank could fire if needed as early as t = 0.

2.4 Stop to Fire Changes

Most of the runs that have been made under this project were made with an attacker who could shoot on the move, and the attacker rarely fired from a stationary position (only when he was mobility but not fire power killed). Some excursions were made into the question of the advantages (if any) of stopping to shoot. Some changes were required in the subroutine HALT which accommodates firing when stopped. The change adds the need to load for the "first" round as mentioned above, and the change also deletes the feature that removes 3 seconds from the laying sequence (which presumably was included since the laying process might have been started before the tank had come to a halt). However it is possible for a tank which is stopped and has just finished firing at a target to sta.. to move and immediately find another target and halt in less tnan 3 seconds. This is exacerbated by the often short times we have been using for the time to lay the first rounds, and it is further complicated by the fact that we have been using times to lay the gun that are different for stationary firing and for moving firing for our baseline fire control system. It is unclear how to pro rate these two laying times during the stopping process. While not thoroughly satisfying, we decided to let the vehicle come to a halt and then use the shorter lay time associated with a stationary firer.

Note that the piece of programming

```
tlast = t - 3.
```

remains as a relic in the modified subroutine.

2.5 Changes for Cardioid Distribution

The original TANKWARS handled the consideration of the azimuthal orientation of hits on the vehicles by creating a distribution of attack geometries. The advancing force proceeded on a course that for each Monte Carlo replication had an orientation that was randomly chosen (usually cardioid) about the axis that joined the center of the target array and the center of the attacking force. Thus the tanks often marched in a direction that resulted in the forces never coming very close. This led to a number of indecisive replications within a sequence of Monte Carlo runs.

AMSAA does not have this feature in GROUNDWARS. The attackers advance toward the target, and they apply a random angle at the time of each impact.

TANKWARS and GROUNDWARS use two tables - a table of dispersions for different cases and ranges and a table of conditional kill probabilities for different kill criteria, dispersions, ranges, exposures, and angles of attack. The probability of hit is calculated using normal distributions for hits on the turret and chassis and then the conditional kill probabilities are multiplied by the hit probability to obtain the probabilities of kill for that instance. These kill probabilities are compared against a random number and one (or none) is selected.

We have adopted a scheme that has the attackers advancing directly toward the defenders as does GROUNDWARS; but we also do a lot of preprocessing of the data to account for the angular distribution of hits. The rationale follows:

The conditional kill tables are large, having some 6000 entries.

The hit probability calculations involve Gaussian function calculations.

We had hopes that we eventually could add more than one type of ammo for each side; so that the memory requirements would get large.

Almost all the shooting was either by stationary vehicles shooting at moving vehicles or vice versa. The issue of the difference between the probability of hit for first and for subsequent rounds in stationary fire against stationary targets was moot (this involves random bias and random dispersion).

The approach was to create a table of hit and kill probabilities (given a shot)as functions of range for:

* the three cases of:
    stationary shooting at stationary targets
    stationary shooting at moving targets
    moving shooting at stationary targets

* two target conditions of:
    hull defilade
    fully exposed

* five levels of kill
    hit
    mobility kill
    fire power kill
    mobility and firepower kill
    catastrophic kill

This along with nine range values (0 to 4000 meters by 500 meters) gives a table with 270 entries and puts a lot of mathematical calculation outside the Monte Carlo replications. The only drawback seems to be the need to do something about the correlation of impacts for the stationary / stationary case should that become important.

The changes to accommodate this kill model are in the subroutines DAMAGE, INPUT, KILL, MAYHIT, and RDPKH. Note that the subroutines ACCERR, ACCMS, ACCSM, ACCSS, IZHIT, and RDEROR are not needed.

Most of the changes are straightforward and relate to shortening the calculation by avoiding specific efforts to calculate hit probability.

One piece of the code is worth discussing. While it is very similar to the original code, it must be understood to understand the BASIC program given below for the calculation of the vulnerability table.

It is in the subroutine KILL:

```
        temp = ranu(C.0)    ranu returns a random number
        IF     (temp .gt. p(1)) THEN
c no hit and no kill
           hit    = .false.
           injury = ALIVE
        ELSEIF (temp .gt. p(2)) THEN
c a hit and a "k" kill
           hit    = .true.
           injury = KKILL
        ELSEIF (temp .gt. p(3)) THEN
c a hit and an "m&f" kill but no "k"
           hit    = .true.
           injury = MFKILL
        ELSEIF (temp .gt. p(4)) THEN
c a hit and an "f" kill but no "m" kill
           hit    = .true.
           injury = FKILL
        ELSEIF (temp .gt. p(5)) THEN
c a hit and an "m" kill but no "f" kill
           hit    = .true.
           injury = MKILL
        ELSE
c a hit but no kill
           hit    = .true.
           injury = ALIVE
        ENDIF.
```

In this section of code the random number (temp) is located within a collection of segments in the unit interval which represent various mutually independent outcomes of the hit or miss:

p(1) is the probability of a hit; so if 1.0 > temp > p(1) the round missed,

p(2) is the probability of any kill less that a K (catastrophic) kill; so if p(1) > temp > p(2) the round hit and achieved a K kill,

p(3) is the probability of any kill that is not both a mobility and a firepower kill; so if p(2) > temp > p(3) the round hit and achieved both a mobility and a firepower kill,

p(4) is the probability of a mobility kill but not a firepower kill; so if p(3) > temp > p(4) the round hit and achieved a firepower kill but not a mobility kill,

p(5) is the probability that the hit produced no kill; so if p(4) > temp > p(5) the round hit but did not kill.

With some thought the reader should be able to convince himself that:

1.0 >= p(1) >= p(2) >= p(3) >= p(4) >= p(5) >= 0.0.

A BASIC program has been written to produce the vulnerability table from accuracy data and from the BRL vulnerability data format. This program considers the turret and the hull to be two shoeboxes. The particular code shown in Appendix B treats both boxes with the same cardioid distribution for all cases (fully exposed, hull defilade, moving, stationary). This can be easily expanded to consider different distributions for different cases, and could be extended to different distributions for the hull and the turret for the same case. The latter would require some rework of the original vulnerability data which only treat the hull and the turret together for one aim point and the turret separately for another aim point. We would need the data for the hull and the turret separately for the one aim point associated with the fully exposed target. Such data are basically available, but probably not in the right form.

The BASIC program was written for and run on a PC. It could of course be easily written in FORTRAN or left in BASIC and run on a more capable computer if desired. The program takes about 10 minutes to run under interpreted BASIC on a Tandy 1000SX. This is no problem since it need run very infrequently.

2.6 Detection and Pinpoint Changes

Changes were made in the two types of detection addressed by TANKWARS - pinpoint detection and the detection of targets without using their firing signatures.

Here, pinpoint detection is defined as the detection of the firing of an opponents gun and the subsequent location of the firing vehicle well enough for the person detecting to be able to aim his gun effectively at the firer. Two probabilistic concepts are involved - the probably that the observer can achieve a pinpoint detection as defined above and the median time it takes him to accomplishment that pinpoint detection.

Pinpoint is handled in the subroutine PINPNT. One change was to allow the median time for the pinpoint process to be input from the appropriate misc input file. TANKWARS has this value as a fixed number and a recompilation of the program was required each time the value was changed. There is a related change in the subroutine RDMISC that gets this median time from the misc file.

We found (as have others) that pinpoint was an important capability for the attacker who is looking for otherwise often hard to find targets that are in hull defilade. One thought for improved systems was to add the capability for a tank to tell other tanks on his side that he saw a firing and give them the location of that targets. To test this idea, we added a change to the subroutine PINPNT that allowed the other tanks to locate the target at the same time that the observer saw it. So far we have been working with this optimistic version of the idea. This could be easily changed in the PINPNT subroutine by adding additional time to the time used to schedule the XFER event; the others would then locate the target later than the original observer.

XFER is the subroutine that has been added to accomplish this transfer.

The misc file contains an additional logical value that is read in the subroutine RDMISC and used to set the value of the variable xxfer which is used as a flag to call or not call transfer in PINPNT.

The other type of detection is what one gets with looking for the targets with binoculars, IR devices, etc. without the aid of a firing signature.

Since the transfer of pinpoint seemed to be a help in some instances and since VHSIC seem to offer considerable possibilities for fire control improvement, we considered the possibility that any target that is detected could be made available to everyone on the side of the observer.

This was accomplished simply enough by adding a change to the subroutine DETECT that is similar to that added to PINPNT. In the version of the program in Appendix A, the variable xxfer either initiates the transfer of both types of detection or inhibits the transfer of any detections.

As alluded to above in Section 1, the availability of all these targets did not produce the Nirvana anticipated. Transfer of pinpoint detections in situations with low pinpoint ability seems desirable, but the gross exchange sometimes seems to produce problems and sometimes seems to help. Also as mentioned above, the prioritization of targets and the ability to keep one target from saturating the attention of the force need much further consideration.

Issues such as, the prioritization of targets, the disengagement algorithms, the ability to transfer targets to vehicles that can't see the targets at that time, and the ability of a moving vehicle to remember stationary targets when the moving vehicle loses line of sight contact all need further consideration. The few limited results we've gotten in this regard have not been satisfactory in the sense of providing understanding. A good bit more work is still needed.

2.7 Changed Handling of Shared Targets

TANKWARS has an option that keeps a tank from firing at a target if another tank is already firing at that target. To accomplish this option, one sets a variable called share() to true to prohibit firing at the same target. We felt that it was extreme to prohibit such firing, but that is was still desirable to be able to limit it. A change was added to the subroutine PRIORN so that if share() = true, the priorities are adjusted so that a target being attacked has a lower priority than anyone not being attacked. The targets being attacked are ordered in the same way among themselves as are the targets not being attacked. Again the results of using this option have been mixed, and more thought about the prioritization scheme is indicated.

2.8 Busy Bug

There is a bug in TANKWARS in the subroutine VANTER that handles the vanishing of vehicles in terrain. If the tank has made the decision to engage a target and then schedules a firing, it has set a flag busy() = true. When the scheduled firing routine is reached, the busy flag is set to false, and a firing-on-target flag is set. If the firing tank loses line of sight by vanishing in terrain before the scheduled firing, VANTER cancels the firing but does not reset the busy flag to false. Since the firing that would have reset the flag has been canceled, the busy flag is never reset during the engagement, and the tank can no longer select any targets after is reappears. We added a fix that sets busy to false in VANTER. This resulted in a sizable improvement in the general performance of the attacking force. The defender has little occasion to use VANTER.

2.9 Know Bug

There is another bug in version 2 of TANKWARS which seems to be caught in version 2a. The variable know() is used in the subroutine PRIORT to distinguish between a target that has been hit and one that has been missed. However, the value know() = 1 was not set in the subroutine MAYHIT if the target was missed (know() = 2 was set if the target was hit). We added a correction for this in MAYHIT.

3. Possible Advantages of GROUNDWARS

GROUNDWARS is more actively maintained and is among other things a newer version of TANKWARS. It also has some additional features including the availability of a very large output option for detailed data on the combat that might be useful for some diagnostics, the ability to handle subgroups of attacking vehicles, the inclusion of the effects of artillery fire, an improved smoke model, and most of all an improved acquisition model which is claimed to produce a factor of six or seven improvement in runtime. This acquisition model replaces the step-by-step model in TANKWARS with a model that predicts when the detections will occur if at all. Note that this is not a different physical model but rather a different mathematical formulation of the solution.

On the other hand, as with TANKWARS, GROUNDWARS also needs an ability to handle leapfrogging, an ability to handle moving meeting engagements, the ability to handle multiple weapon types, and some rework of the assignment of priorities for engagement and disengagement. Also Dr. Howe wishes to look at the synergistic effects among multiple hits on a target which will require a new vulnerability model for either TANKWARS or GROUNDWARS.

4. What Next

The next major step that is planned is to develop a more complete model which can handle combined arms in essentially the same detail as TANKWARS or GROUNDWARS and address the additional issues mentioned above. This could be a natural (but very large) step forward from developing the ability to have more than one armored weapon system on each side, the ability to do leapfrogging, and the ability for handling more dynamic meeting engagements.

While there is that longer term goal, there is also a shorter term set of questions to which Dr. Howe needs answers which include:

* The ability to model advanced C3 concepts including the ability to remember targets, the ability to transfer targets, and a rework of the assignment of priorities for engagement and disengagement.

* A new vulnerability model that can account for possible synergistic effects among impacts on the target.

* The ability to handle a meeting engagement between two forces that are moving.

* The ability to have more than one weapon systems on each side. In particular, the ability to have both missiles and guns on each side.

There are two interesting approaches to the longer term goal:

(1). to start from scratch and build a model using the "C" programming language and particularly using "structures" to represent the fundamental elements (weapon systems, vehicles, etc.)

(2). to build on existing code (here it would seem desirable to use GROUNDWARS, specially in view of the new version 4.0, it's state of maintenance, and it's acceptance by AMSAA.). Note here that this would most certainly require major surgery.

CONCLUSIONS

It's interesting that both AMSAA people and the author arrived at the conclusion that structures were particularly appropriate in this application. AMSAA is presently developing yet a newer combat model using "C". In terms of the final (if one could say that there ever is a final version of such a code) code, the use of structures would be the more powerful approach and would be more amenable to the modifications that are always required to consider new weapon and tactical concepts. The difficulty, of course, with such an approach is that there will be little short term benefit of such an effort. And the questions to which the above changes could produce answers would have to wait.

The second approach has the advantage that the things to be added to the code could produce useful results as they are added. However, where major surgery is necessary this will not be so likely. On the other hand, the use of FORTRAN and the basic architecture of GROUNDWARS which relies on numerous globally declared arrays could make generalization of the code very difficult and could make the resulting code very opaque and very awkward to adapt to changing concepts.

It is the author's opinion that the first approach is the proper one. At the same time it may be possible to provide some short term ability without too much difficulty by adding a new prioritization scheme and a more complete vulnerability model to existing code. The exercise of particularly the prioritization scheme might be valuable in the development of a new code by providing a mechanism for discussing tactics with the User with a view toward a better representation of tactics in the new code.

Certainly even if the first approach is taken, TANKWARS and GROUNDWARS will valuable sources of ideas.

REFERENCES

1. Fred Bunn, Unpublished paper on TANKWARS, if anyone is interested in TANKWARS, he should contact Mr. Bunn at the Ballistic Research Laboratory (301-278-6676).

2. Michael C. Schmidt, Gary R. Comstock, Lilly D. Harrington, Barry J. Burns, "GROUNDWARS 4.0 User's Guide", AMSAA Technical Report, October 1989

APPENDIX A

TANKWARS PROGRAM

INTENTIONALLY LEFT BLANK.

Appendix A
TANKWARS Program

```
c V%i%
c       common.h file
c       -------------
c       common block declarations for Extended Combat Simulation (TankWars II)

        implicit integer(i-n), real(a-h,o-z)

        parameter (NN=20)
        character*4 color
        character*1 kview
        integer ALL, NULL, FLS TGT
        integer FD, HD, FE
        integer TURRET, HULL
        integer BLU, RED
        integer MEETNG, RATTAK, BATTAK
        integer ALIVE, MKILL, FKILL, MFKILL, IKILL, KKILL
        integer SLOWNG, STATNY, ACCELG, MAXVL
        integer scene, tactic, prevrd, army
        logical cansee, fot, mot, kncels
        logical busy, empty, foes, los, seen, serchg, repeat
c      Change by HLReed
        logical istest, share, xxfer, memory
        real INFINT
c       Change introduced by H.L.Reed on 8 Mar 89 to allow overwatch
c       tanks to be added to the attacking force.  See also changes in
c       subroutines cango, deplo2, init2, and input.
        logical inwatch
c
        common /aspekt/ angles(15), pangle(15), iangd
        common /charc / color(2), kview(2)
        common /consts/ PI, TWOPI, DEG, VNORTH(3)
        common /const2/ ALL, NULL, FLS TGT,
     1     FD, HD, FE, TURRET, HULL, BLU, RED, MEETNG, RATTAK,
     2     BATTAK, ALIVE, MKILL, FKILL, MFKILL, IKILL, KKILL ,
     3     SLOWNG,STATNY,ACCELG,MAXVL,INFINT
        common /contrl/ nreps, keyd(20), keym(20), scene, tmax, meth sm
        common /cpath / nmaxt(2),accel(2),decel(2),ishtfs(2),
     1     speed(2), angle(2), accmax(2), wvlth(2), ampl(2)
        common /crandm/ irandm, jrandm
        common /cshot/  kshot(2,20)
        common /ctrace/ trace
        logical         trace
        common /endgam/ sysdim(2,8), nang, ndisp
c       common /errors/ ssrgs(2,10), smrgs(2,10), velms(2,20),
c     1    sserrs(2,16,10), smerrs(2,16,10), addons(2,2,20), nadds(2)
        common /error2/ rex, rey, reliab(2)
c Change by HL Reed to allow transfer added xxfer.  Also added the
c   variables which follow xxfer but are not being used at this time.
        common /fcycle/ nrds(2), nrpt(2), nipods(2), nrpb(2), tactic(2),
     1     tof(2,8),trelod(2), tfirst(2,8), tmedin(2), tfixed(2,8),
     2     rof(2),kind rd(2),tbump(2),nbump(2),thide(2),tmin(2),
     3     nprior(2),nchans(2),share(2),xxfer(2),memory(2),
     4     tpop1(2),tpop2(2),tpop3(2),nstatn,nmove,rstatn,
     5      rmove, ttotal, nstatf, nmovef, rstatf, rmovef
        common /n sys/ ntanks(4,6), nblu, nred
c       Change by HLReed added pntime for median pinpoint time
        common /sensor/ psense(2,8), pinfin(2,3,10), tbar(2,3,10),
     1     ndets(2), tlook(2), pinp(2), repeat, recknz(2), pfalse(2,2),
     2     pntime(2)
        common /states/ army(NN), aspect(NN,2), cansee(NN),
     1  busy(NN), empty(NN), fot(NN,NN), foes(NN,NN), ichg(NN),
     2  knceal(NN), kncels(NN,NN), know(NN,NN),
     2  life(NN), los(NN,NN), mot(NN,NN), motion(NN), mslfly(2,NN,5),
     3  nhot(NN), nbrst(NN), ndet(NN), nrd(NN), nrib(NN), nipod(NN),
     4  nrot(NN), nrtgt(NN), nchan(NN), prevrd(NN), rgvis(3,NN),
     5  seen(NN,NN), serchg(NN), tfire(NN,NN), tfire2(NN), vbx(NN),
     6  vby(NN), t0(NN), x0(NN), y0(NN), vx0(NN), vy0(NN),
     7  xp(NN), tlast(NN)
        common /state2/ idecoy(NN), iflash(NN), ndecoy(2), nflash(2)
```

```
c       Change for output  HLReed
        common /stats/ statb(8), mystat(4)
        common /tstore/ a(1000), iholy
        common /vars6/ irginc, rgincr, rginc2
        common /csmoke/ tsmoke(20),psmoke(20,3), invisb
        common /smokel/ toutil(21,5),toutvl(21,5),touti(21,5),
     1    toutv(21,5),tini(21,5),tinv(21,5),ptbl(21),rtbl(5)
        common /v16a/ krep
        common /v17a/ istest
        common /v23/ nwave, nwaves, nsurv, neval, nused(3000), nreps3,
     1  statc(8), noammo, loammo, noamo2, loamo2
        common /v24/ nstats(5,2)
        common /where/ min rg, max rg, inc rg
        common /where2/ nrg, rg0, rg, s(3), vt(3), vf(3)
        common /fitnes/ quit(2,2), alloc(2,6), fit(NN,6)
c       Change introduced by H.L.Reed on 8 Mar 89 for overwatch
        common /ovrwtch/ nwatch(3),inwatch(NN)

c V7.2
c       MAIN ROUTINE
c 9     Main: read input and simulate scenarios.
        include 'common.h'
1       format (' The Sustained Combat Model: Tank Wars II.',/,
     1  ' Written by Fred Bunn (ph (301) 278-6648, autovon 298-6648)',/,
     2  ' Ballistic Research Laboratory, Aberdeen Proving Ground, MD',/,
     3  ' Version 7.2  Created 10/24/88.',/)
c
        call input
        DO 20 scene=1,3
          call forces
20      CONTINUE
        END
c V7.1
```

```
       SUBROUTINE ABORT (t,firer,tgt)
c 6    Abort: abort msl from firer to tgt (to all tgts if tgt=0)
       include 'common.h'
       logical defndr
       integer armyf, armyt, firer, tgt
1      format(f8.2,1x,a4,i3,' msl for  ',a4,i3,' aborted.')
2      format ('ABORT: firer,tgt,i,mslfly=',4i3)
3      format ('ABORT: msl approaching tgt',i3)
c
       if (trace) print *,'>abort'
       armyf = army(firer)
       armyt = 3-armyf
       DO 20 i=1,5
c      Check all 5 missile pointers for this firer
         msl = msl fly(armyf,firer,i)
         if (keym(19).gt.0) print 2, firer, tgt, i, msl
         IF (msl.gt.0) THEN
c        Missile found (pointer is non-zero)
           msl tgt = a(msl+1)
           if (keym(19).gt.0) print 3, msl tgt
           IF (tgt.eq.0 .or. tgt.eq.msl tgt) THEN
c            Abort this missile
             kshot(armyf,3) = kshot(armyf,3)+1
             call cancel (msl, eIMPCT, NULL)
             msl fly(armyf,firer,i) = 0
             if (msl tgt.ne.FLS TGT) mot(firer,msl tgt) = .false.
c            Release area for storage of missile data
               a(msl) = -a(msl)
               if (keyd(1).ge.2) print 1,t,color(armyf),firer,
     1           color(armyt),msl tgt
c            Pop-down to reload if defender, pod empty, & fully alive
               defndr = (scene.eq.BATTAK .and. armyf.eq.RED) .or.
     1           (scene.eq.RATTAK .and. armyf.eq.BLU)
               if (defndr.and.empty(firer).and.(life(firer).le.ALIVE))
     2           call skedul(t,firer,ePOPDN,NULL)
           ENDIF
         ENDIF
20     CONTINUE
       if (trace) print *,'<abort'
       END
c V7.1
```

```
      SUBROUTINE ACCELF (t, firer)
c 9   Accelf: simulate tank starting to accelerate.
      include 'common.h'
      integer firer
1     format (f8.2,1x,a4,i3,' speed up',9x,'(was slowing)')
2     format (f8.2,1x,a4,i3,' speed up',9x,'(was halted)')
3     format (f8.2,1x,a4,i3,' speed up',9x,'(was speeding up)')
4     format (f8.2,1x,a4,i3,' speed on',9x,'(is cruising)')
c
      if (keyd(4).gt.0) print *,'>accel'
      if(life(firer).ne.FKILL.and.invisb.eq.1.and.knceal(firer).ne.FD)
     1  call skedul (t,firer,'vanish',NULL)
      narmy = army(firer)
      IF (motion(firer).eq.SLOWNG) THEN
c     Previous motion was slowing
        if (keyd(1).ge.2) print 1, t, color(narmy), firer
        call path(firer,t,motion(firer),0.0,x,y,vx,vy)
        dt = (speed(narmy)-vy)/accel(narmy)
        call skedul (t+dt,firer,'maxvel',NULL)
        motion(firer) = ACCELG
      ELSE IF (motion(firer).eq.STATNY) THEN
c     Previous motion was stationary
        if (keyd(1).ge.2) print 2, t, color(narmy), firer
        call path(firer,t,motion(firer),0.0,x,y,vx,vy)
c       schedule time full velocity reached (max vel)
          dt = speed(narmy)/accel(army(firer))
          call skedul(t+dt,firer,'maxvel',NULL)
          motion(firer) = ACCELG
      ELSE IF (motion(firer).eq.ACCELG) THEN
c     Previous motion was accelerating
        if (keyd(1).ge.2) print 3, t, color(narmy), firer
      ELSE IF (motion(firer).eq.MAXVL) THEN
c     Previous motion was cruising at max velocity
        If (keyd(1).ge.2) print 4, t, color(narmy), firer
      ENDIF
      if (keyd(4).gt.0) print *,'<accel'
      END
c V7.1
```

```
      FUNCTION ANGLEF (a, b)
c 9   Anglef: find angle between two vectors.
      dimension a(3), b(3)
c
      vabsa = sqrt( dot( a,a ) )
      vabsb = sqrt( dot( b,b ) )
      dotab =       dot( a,b )
CHANGED 1 Apr 86. Next line replaced by 3.
c     dm = acos(dotab/(vabsa*vabsb))
      dm = dotab/(vabsa*vabsb)
      dm = amin1(1.,amax1(-1.,dm))
      dm = acos(dm)
      r3 = a(1)*b(2) - a(2)*b(1)
      anglef = -sign(dm,r3)
      END
c V7.1
```

```
        FUNCTION ANGSUM (a, b)
c 3     Angsum: add 2 angles and adjust answer to lie between +-PI.
        c = a+b
10      IF (c.lt.-180.) THEN
          c = c+360.
          GOTO 10
        ELSE IF (c.gt.180.) THEN
          c = c-360.
          GOTO 10
        ENDIF
c         angle is adjusted.
        angsum = c
        END
c V7.3
```

```
        SUBROUTINE APPEAR(t,tgt,firer)
c 0     Appear: if tgt appears treat, otherwise reschedule appearance
        include 'common.h'
        integer tgt,firer, armyf, armyt
        common /terane/ d(40), xold(20), yold(20), dist(20), iseg(20)
        rss(x,y) = sqrt(x*x+y*y)
1       format(f8.2,1x,a4,i3,' appears ',9x,'(x=',f8.1,'   y=',f8.1,')')
2       format(f8.2,1x,a4,i3,' LOS to   ',a4,i3,' starts.')
c
        if (trace) print *,'>appear'
        armyt = army(tgt)
        armyf = 3-armyt
        IF (invisb.eq.1) THEN
          if(speed(armyt).le.0.)print *,'APPEAR: armyt,speed=',armyt,
     1      speed(armyt)
          if(speed(armyt).le.0.) stop
          call path(tgt,t,motion(tgt),0.2,x,y,vx,vy)
c       Terrain causes intermittent LOS.
          travel = rss(x-xold(tgt), y-yold(tgt))
          IF (travel.gt.dist(tgt)) THEN
c         Tgt is no longer masked by terrain
            if (keyd(1).gt.1) print 1,t,color(armyt),tgt,x,y
            xold(tgt) = x
            yold(tgt) = y
            iseg(tgt) = iseg(tgt)+1
            if (iseg(tgt).gt.40) iseg(tgt)=iseg(tgt)-40
            dist(tgt) = d(iseg(tgt))
            call aprter(t,tgt,firer,FE)
c           Schedule next disappearance
              dt = dist(tgt)/speed(armyt) + 0.01
              call skedul(t+dt,tgt,'vanish',NULL)
          ELSE
c         Still masked by terrain, so reschedule mask end
            IF (life(tgt).eq.ALIVE) THEN
              dt = (dist(tgt) - travel) / speed(armyt) + 0.01
              call skedul (t+dt,tgt,'appear',NULL)
            ENDIF
          ENDIF
        ELSE
c       Tgt is no longer masked by smoke
          if (keyd(1).gt.1) print 2,t,color(3-armyt),firer,
     1      color(armyt),tgt
          call aprsmk(t,tgt,firer)
c         Schedule next disappearance
            r = rgf(t,firer,tgt)
            p = ranu(0)
            pout = ranu(0)
            IF (kview(RED).eq.kview(BLU)) THEN
              IF (armyf.eq.BLU) THEN
                IF (kview(armyf).eq.'I') THEN
                  dtin=tdintp(ptbl,rtbl,tini,p,r,21,5)
                  dtout=tdintp(ptbl,rtbl,touti,pout,r,21,5)
                ELSE
                  dtin=tdintp(ptbl,rtbl,tinv,p,r,21,5)
                  dtout=tdintp(ptbl,rtbl,toutv,pout,r,21,5)
                ENDIF
                call skedul(t+dtin,tgt,'vanish',firer)
                call skedul(t+dtin,firer,'vanish',tgt)
                call skedul(t+dtin+dtout,tgt,'appear',firer)
                call skedul(t+dtin+dtout,firer,'appear',tgt)
              ENDIF
            ELSE
              IF (kview(armyf).eq.'V') THEN
                dtinv=tdintp(ptbl,rtbl,tinv,p,r,21,5)
                dtoutv=tdintp(ptbl,rtbl,toutv,pout,r,21,5)
                dtouti=tdintp(ptbl,rtbl,touti,pout,r,21,5)
                dtini=dtinv+(dtoutv-dtouti)*0.5
                call skedul(t+dtinv,tgt,'vanish',firer)
                call skedul(t+dtinv+dtoutv,tgt,'appear',firer)
              call skedul(t+dtini,firer,'vanish',tgt)
              call skedul(t+dtouti+dtini,firer,'appear',tgt)
            ENDIF
          ENDIF
        ENDIF
      if (trace) print *,'<appear'
      END
c V7.1
```

```
      SUBROUTINE APRSMK(t,tgt,firer)
c 0   Aprsmk: Tgt appears out of smoke, reset.
      include 'common.h'
      integer tgt,firer
      common /terane/ d(40), xold(20), yold(20), dist(20), iseg(20)
c
      if (trace) print *,'>apprsmk'
      narmy = army(tgt)
c     Restore all lines-of-sight involving tgt
        los(firer,tgt) = army(firer).ne.narmy
c     Turn search on if it is off
      IF (.not.repeat) THEN
        repeat = .true.
        call skedul(t+.01,0,'search',NULL)
      ENDIF
      if (trace) print *,'<aprsmk'
      END
c V7.1
```

```
      SUBROUTINE APRTER(t,tgt,firer,jexpos)
c 0   Apprter: Tgt has appeared from behind terrain, reset.
      include 'common.h'
      integer tgt,firer
      common /terane/ d(40), xold(20), yold(20), dist(20), iseg(20)
1     format(f8.2,1x,a4,i3,' aprters ',9x,'(x=',f8.1,'   y=',f8.1,')')
c
      if (trace) print *,'>aprter'
      naray = aray(tgt)
      knceal(tgt) = jexpos
c     Restore all lines-of-sight involving tgt
       DO 20 i=1,nblu+nred
        IF (knceal(i).ne.FD) THEN
         los(tgt,i) = aray(i).ne.naray
         los(i,tgt) = aray(i).ne.naray
        ENDIF
20     CONTINUE
c     Turn search on if it is off
       IF (.not.repeat) THEN
        repeat = .true.
        call skedul(t+.01,0,'search',NULL)
       ENDIF
      if (trace) print *,'<aprter'
      END
c V7.1
```

```
      BLOCK DATA BLKDAT
      include 'common.h'
      data color,           pi,          twopi,       deg
    1    /'Blue', 'Red ', 3.141592654, 6.283185308, 57.29577951/
      data VNORTH /0., 1., 0./
      data ALL, NULL, FLS TGT /0, 0, -1/
      data FD, HD, FE /1, 2, 3/
      data TURRET, HULL /1, 2/
      data BLU, RED /1, 2/
      data MEETNG, RATTAK, BATTAK /1, 2, 3/
      data ALIVE, MKILL, FKILL, MFKILL, IKILL, KKILL /1,2,3,4,5,6/
      data    SLOWNG, STATNY, ACCELG, MAXVL,  INFINT
    1 /       1,      2,      3,      4,      1.e35/
      data keyd, keym /40*0/
      END
c V7.1
```

```
      SUBROUTINE BOUNDS (narmy, box, angll, r1, r2)
c  8  Bounds: find the horizontal bounds of hull or turret.
c     Definitions:
c       angll - angle off the nose of the box (rad).
c       box - 1 means turret box, 2 means hull box.
c       narmy - 1 means blue firers, 2 means red firers.
c       c, s2, s3 - temporary variables.
c       r1, r2 - left and right boundaries of boxes (m).
      include 'common.h'
      integer box
c
      if (trace) print *,'>bounds'
c     initialize
        temp = (angll+twopi)/twopi
        theta= (temp-aint(temp))*twopi
c       theta = amod (angll+twopi,twopi)
        c = sysdim(narmy,4*(box-1)+2) * cos(theta)
        s2= sysdim(narmy,4*(box-1)+3) * sin(theta)
        s3= sysdim(narmy,4*(box-1)+4) * sin(theta)
c
      IF (theta.le.0.25*twopi) THEN
c     case 0 < theta <= 90
        r1 = -s2 - c
        r2 =  s3 + c
      ELSEIF (theta.le.0.5*twopi) THEN
c     case 90 < theta <= 180
        r1 = -s2 + c
        r2 =  s3 - c
      ELSEIF (theta.le.0.75*twopi) THEN
c     case 180 < theta <= 270
        r1 =  s3 + c
        r2 = -s2 - c
      ELSE
c     case 270 < theta <= 360
        r1 =  s3 - c
        r2 = -s2 + c
      ENDIF
      if (trace) print *,'<bounds'
      END
c V7.1
```

```
      SUBROUTINE CANCEL (I, act, it)
c 0   Cancel: cancel 'act' events for 'I' entity.
c       (all events if act='')
c     Definitions of local variables:
c       m - pointer to previous event
c       n - pointer to current event being considered
      include 'clock.h'
      logical is what, is who, is whom
      character*6 act
1     format(9x,'cancel ',i3,' ',a6,i3,' at time',f8.2)
c
      m = 0
      n = nxevnt
10    IF (n.ne.0) THEN
c     Continue until n=0
        is who  = I .eq.who(n)
        is what = act.eq.what(n) .or. act.eq.'all   '
        is whom = it.eq.whom(n) .or. it.eq.0
        IF (is who .and. is what .and. is whom) THEN
c       Then remove event
          if (prflag )print 1, I, act, it, when(n)
          if (m.eq.0) nxevnt = next(n)
          if (m.ne.0) next(m) = next(n)
          next(n) = nxidle
          nxidle = n
          if (m.eq.0) n = nxevnt
          if (m.ne.0) n = next(m)
        ELSE
c       Don't remove event. Shift to next event.
          m = n
          n = next(n)
        ENDIF
        GOTO 10
      ENDIF
      END
c V7.1
```

```
      LOGICAL FUNCTION CAN GO (firer, t)
c 6   Can go: True iff is stationary and can move.
      include 'common.h'
      integer firer
      logical is atkr, m alive, faster
c
      narmy = army(firer)
      is atkr = (narmy.eq.BLU .and. scene.eq.BATTAK) .or.
     1  (narmy.eq.RED .and. scene.eq.RATTAK)
      m alive = life(firer).eq.ALIVE  .or.
     1  life(firer).eq.FKILL
      faster = (motion(firer).eq.STATNY .or.
     1  motion(firer).eq.SLOWNG)
c     Change introduced by H.L.Reed on 8 Man 89 to allow overwatch tanks
c     to be added to the attacking force.  See also changes in subroutines
c     deplo2, init2,  and input and in common.h.
      can go = is atkr .and. m alive .and. faster .and.
     1 (.not. inwatch(firer))
      END
c V7.1
```

```
      SUBROUTINE CREATE (n, ient)
c 8   Create: create a temporary entity. (a bullet or msl)
c     purpose - this routine 'creates' a temporary entity.  what it
c     actually does is find space to store the attributes of that
c     entity and reports back the index of the first storage word
c     as the entity number.
c     definitions -
c     a       is the vector used to store attributes in.
c     i       this is the index of the storage space we are currently
c        looking at.
c     ient    is the index of the storage space where the attributes
c        will be stored.  it is also the number that will be used
c        to identify the temporary entity created.
c     isdone  the routine is done if isdone = 1 and it is not done
c        if isdone = 0.  the only reason for the routine to be done
c        is if it finds space to store the attributes in.
c     istart  this is the starting point for the search.  if we get
c        back to istart without a find, we have a storage overload
c        and we error off.
c     j       this is the index of the next storage space.  we want
c        to look at it with the possibility of catenating it to the
c        storage space beginning at i.c        n       the number of
c        attributes to be stored.
c     nreq    this is the number of spaces required.  it equals the
c        number of attributes plus one word.  this one word is used
c        for searching purposes.  if it is negative (-abs(m)), that
c        indicates that the next m words are being used to store
c        the m attributes of an entity.  if it is positive, then the
c        next m words are available for use.
c    note - the amount of storage space is 1000 words.  if you want to
c      increase this you'll have to change all occurences of 1000.
c      also note that in initx these must be set - a=0, a(1)=1000.,
c      i=1.
c
      logical trace
      common /contrl/ nreps, keyd(20), keym(20), scene, tmax, meth sm
      common /ctrace/ trace
      common /tstore/ a(1000), i
1     format (10x,'CREATE:  Not enuf space to store',
     1 i5, ' attributes.')
2     format (10x,'CREATE: i, j, a(i), a(j) =',
     1  /10x,2i5,2f10.3)
c
      if (trace) print *,'>create'
c     Initialize
      isdone = 0
      istart = i
      nreq = n+1
c     Find empty space in the a-array
10    IF (isdone.ne.1) THEN
c     Try next empty space
c       Catenate empty spaces if possible
 20     CONTINUE
c         Find next space (and error off if we're back at start)
          j = i+iabs(int(a(i)))
          if (j.gt.1000) j=1
          IF ((j.eq.1) .or. (a(i).lt.0) .or. (a(j).lt.0)) THEN
c         test this space for size.
          IF (a(i).lt.float(nreq)) THEN
c           move to next space
            i = j
            if (i.eq.istart) print 1, n
            IF (i.eq.istart) STOP
          ELSE
c           reserve space
            isdone = 1
            itemp = i+nreq
              if(a(i).ne.float(nreq))a(itemp) = a(i)-float(nreq)
            a(i) = -nreq
            ient = i
            i = j
          ENDIF
        ELSE
c         do catenation
          a(i) = a(i)+a(j)
          IF (a(i).gt.0.0 .and. a(j).gt.0.0) GOTO 20
          print 2, i, j, a(i), a(j)
          STOP
        ENDIF
        GOTO 10
      ENDIF
      if (trace) print *,'<create'
      END
c V7.1
```

```
      SUBROUTINE CRESET
c 0   Creset - Reset variables used by create.
      common /tstore/ a(1000), iholy
      parameter (NN=20)
      DO 20 i=2,1000
        a(i)=0.0
20    CONTINUE
      a(1)=-NN
      a(NN+1) = 1000-NN
      iholy=NN+1
      END
c V7.1
```

```
        SUBROUTINE DAMAGE (t, I, it, injury)
C 0     Damage: schedule effects.
c       Changed May 18, 1989 for simplified hit and kill model, HL Reed
        include 'common.h'
        character*2 kt(6)
        data kt /'no','M-','F-','MF','I-','K-'/
1       format(f8.2,1x,a4,i3,1x,'Hits     ',a4,i3,' (no damage).')
2       format(f8.2,1x,a4,i3,1x,a2,'-kills ',a4,i3)
c
        if (trace) print *,'>damage'
        n=array(I)
        m = 3-n
          IF(keyd(1).ge.2) THEN
            if (injury.eq.1) print 1,t,color(n),I,color(m),it
            if (injury.gt.1) print 2,t,color(n),I,kt(injury),color(m),it
          ENDIF

        injold = life(it)
        IF (injury.eq.KKILL .and. injury.ne.injold) THEN
c       Treat first catastrophic kill.
          life(it) = KKILL
          call damagf(t,it,m)
          call damagm(t,it)
          call cancel(it,'ikill ',NULL)
          call newtgt (t,I,it)
          call deaths(t)
        ELSEIF (injury.ne.injold .and. injold.lt.MFKILL) THEN
c       Treat new damage (less than catastrophic).
          IF (injury.eq.MKILL) THEN
            if (injold.eq.FKILL) life(it) = MFKILL
            if (injold.eq.ALIVE) life(it) = MKILL
            if (injold.eq.ALIVE .or. injold.eq.FKILL) call damagm (t,it)
          ELSE IF (injury.eq.FKILL) THEN
            if (injold.eq.ALIVE) life(it)=FKILL
            if (injold.eq.MKILL) life(it)=MFKILL
            call damagf(t,it,m)
          ELSE IF (injury.eq.MFKILL) THEN
            if (injold.lt.MFKILL) life(it) = MFKILL
            if (injold.ne.MKILL) call damagm (t, it)
            if (injold.ne.FKILL) call damagf (t,it,m)
          ENDIF
          if (life(it).eq.MFKILL.and.injold.lt.MFKILL)
     1      call skedul(t+tbump(n),it,'ikill ',NULL)
        ENDIF
        if (trace) print *,'<damage'
        END
c V7.1
```

```
        SUBROUTINE DAMAGF (t,it,m)
c       Damagf - Discard activities due to firepower kill.
        include 'common.h'

        nrtgt (it) = 0
        nchan(it) = 0
c       Clear any guidance channels in use by target.
          if (kindrd(m).eq.4) call abort(t,it,0)
          DO 40 j=1,nblu+nred
            fot(it,j) = .false.
40        CONTINUE
        call cancel(it,'fire  ',NULL)
        call cancel(it,'reload',NULL)
        call cancel(it,'select',NULL)
        IF (life(it).eq.FKILL .and. speed(m).gt.0.0) THEN
          call cancel (it,'slowup',NULL)
          call cancel (it,'halt  ',NULL)
          call cancel (it,'accel ',NULL)
          call skedul (t, it, 'accel ',NULL)
          dt = thide(m)
          if (m.eq.BLU .and. scene.eq.RATTAK) dt = 5.0
          if (m.eq.RED .and. scene.eq.BATTAK) dt = 5.0
          call skedul (t+dt,it,'hide  ',NULL)
        ENDIF
        END
c V7.1
```

```
      SUBROUTINE DAMAGM (t, it)
c 9   Damagm - Simulate mobility kill on the tgt.
      include 'common.h'
      logical sos
c     sos - stopped or slowing
      if (trace) print *,'>damagm'
      call cancel (it, 'maxvel', NULL)
      call cancel (it, 'accel ', NULL)
      call cancel (it, 'hide  ', NULL)
      sos = vabs(vt).le.0.0   .or.   motion(it).eq.SLOWNG
      if (.not.sos) call skedul (t, it, 'slowup', NULL)
      if (trace) print *,'<damagm'
      END
c V7.1
```

```
      SUBROUTINE DEATHS (t)
c 0   Deaths: Find death toll on each side. A tank is considered
c     dead if it is I-killed, K-killed, or F-killed & hidden.
      include 'common.h'
      logical dead1, dead2
      integer dead(2)
1     format (i3,' Blu dead,',i3,' Red dead.')
c
      if (trace) print *,'>deaths'
      dead(BLU) = 0
      dead(RED) = 0
c     Change made by H.L. Reed on March 31, 1989 to keep decoys out
c     of the win decision and to keep them out of the exchange ratio
c     (see also finish.f).
      DO 20 i=1,(nblu-ndecoy(BLU))
        dead1 = life(i).ge.IKILL
        dead2 = knceal(i).eq.FD .and. life(i).ge.FKILL
        if (dead1 .or. dead2) dead(BLU)=dead(BLU)+1
20    CONTINUE
      DO 30 i=nblu+1,(nblu+nred-ndecoy(RED)-nwatch(scene))
        dead1 = life(i).ge.IKILL
        dead2 = knceal(i).eq.FD .and. life(i).ge.FKILL
        if (dead1 .or. dead2) dead(RED)=dead(RED)+1
30    CONTINUE
      if (keyd(1).ge.2) print 1,dead
      if ((nblu-ndecoy(BLU)).eq.dead(BLU) .or.
     1 (nred-ndecoy(RED)-nwatch(scene)).eq.dead(RED))
     1   call skedul(t+5.,NULL,'finish',NULL)
      if (trace) print *,'<deaths'
      END
c V7.1
```

```
      SUBROUTINE DEPLO2(isouth,jsouth,nsouth,inorth,jnorth,nnorth)
c 0   Deploy: position & orient all tanks at beginning of engagement.
      include 'common.h'
2     format('tank',i2,' x=',f6.0,' y=',f6.0,' heading=',f5.0,
     1  ' speed=',f5.1,'m/s')

      if (trace) print *,')deplo2'
      spacng = 100.0
c     Position southern tanks on the x-axis
        vsouth = 0.0
        if (scene.eq.BATTAK) vsouth = speed(BLU)
        if (scene.eq.RATTAK) vsouth = speed(RED)
        x0(isouth) = -0.5*(nsouth-1)*spacng
        xp(isouth) = x0(isouth)
        y0(isouth) = 0.0
        vy0(isouth) = vsouth
        aspect(isouth,TURRET) = 0.0
        aspect(isouth,HULL) = 0.0
        n = isouth
c     Related to 8 Mar 89 change
        inwatch(n) = .false.
        if (keyd(1).ge.2) print 2, n, x0(n), y0(n), 0.0, vy0(n)
        DO 20 n=isouth+1,jsouth
          x0(n) = x0(n-1) + spacng
          xp(n) = x0(n)
          y0(n) = 0.0
          vy0(n) = vsouth
c     Change introduced by H. L. Reed on 8 Mar 89 to allow overwatch
c     tanks to be added to the attacking force.  See also changes in
c     the subroutines cango, init2, and input and in common.h.
          inwatch(n) = .false.
          IF(n.gt.(jsouth - nwatch(scene))) THEN
           inwatch(n) = .true.
           vy0(n) = 0.0
           motion(n) = STATNY
           knceal(n) = HD
          ENDIF
c     End of 8 Mar 89 change
          aspect(n,TURRET) = 0.0
          aspect(n,HULL) = 0.0
          if (keyd(1).ge.2) print 2, n, x0(n), y0(n), 0.0, vy0(n)
20      CONTINUE
c     Position northern tanks
c       find center of northern line of tanks
c     Change by HLReed  theta1 is no longer used to provide the cardioid
c     distribution.  That is done in the new vulnerability model.
          theta1 = 0.
          xcen = rg0*sin(theta1)
          ycen = rg0*cos(theta1)
c       place northern tanks on northern line
c     Change by HLReed  theta2 is no longer used to provide the cardioid
c     distribution.  That is done in the new vulnerability model.
          theta2 = 0.
          headng = theta1+theta2+PI
          cosa = cos(headng)
          sina = sin(headng)
          dx = -cosa*spacng
          dy =  sina*spacng
          x0(inorth) = xcen-0.5*(nnorth-1)*dx
          y0(inorth) = ycen-0.5*(nnorth-1)*dy
          vy0(inorth) = 0.0
          dm = headng*deg
          n = inorth
          if (keyd(1).ge.2) print 2, n, x0(n), y0(n), dm, vy0(n)
CHANGED 31 Mar 85 Following 2 lines added.
          aspect(inorth,TURRET) = headng
          aspect(inorth,HULL) = headng
          DO 30 n=inorth+1,jnorth
c      Related to 8 Mar 89 change
            inwatch(n) = .false.
            x0(n) = x0(n-1) + dx
            y0(n) = y0(n-1) + dy
            vy0(n) = 0.0
            aspect(n,TURRET) = headng
            aspect(n,HULL) = headng
            dm = headng*deg
            if (keyd(1).ge.2) print 2, n, x0(n), y0(n), dm, vy0(n)
30        CONTINUE
      last=nnorth+nsouth
      if (trace) print *,'<deplo2'
      END
c V7.1
```

```
      SUBROUTINE DEPLOY
c 0   Deploy: position & orient all tanks at beginning of engagement.
      include 'common.h'

      if (trace) print *,'>deploy'
      DO 20 n=1,nblu+nred
          t0(n) = 0.0
20    CONTINUE
      IF (scene.eq.BATTAK) THEN
c     position blue tanks on the x-axis
          call deplo2(1,nblu,nblu,nblu+1,nblu+nred,nred)
      ELSE
c     position red tanks on the x-axis
          call deplo2(nblu+1,nblu+nred,nred,1,nblu,nblu)
      ENDIF
      if (trace) print *,'<deploy'
      END
c V7.1
```

```
      SUBROUTINE DETECT (t, firer, tgt)
c 3   Detect: find if tgt detected and schedule subsequent events.
      include 'common.h'
      integer firer, tgt, armyf, armyt
1     format (f8.2,1x,a4,i3,' detects ',1x,a4,i3)
c
      if (trace) print *,'>detect'
      armyf = army(firer)
      armyt = 3-armyf
      IF (los(firer,tgt) .and. .not.seen(firer,tgt) .and.
     1  ndet(firer).lt.ndets(armyf)) THEN
        if(keyd(1).ge.2)print 1,t,color(armyf),firer,color(armyt),tgt
        ndet(firer) = ndet(firer)+1
        seen(firer,tgt) = .true.
c     Set thuman to zero as does GROUNDWARS  -  HLReed
        t human = 0.0*exp(rolln(0.5))
        call selecs(t,firer,thuman)
c     Change by HLReed to allow transfer of targets
        IF(xxfer(armyf)) THEN
          i = 1
          if(firer .GT. nblu) i = nblu+1
          call skedul(t+2.,i,'xfer  ',tgt)
        ENDIF
      ENDIF
      if (trace) print *,'<detect'
      END
c V7.3
```

```
      SUBROUTINE DET RG (narmy)
c     Det rg: Find the max ranges at which each firer in 'narmy' detects.
      include 'common.h'
      integer narmy, first, last, tank, cond, krg
      real p1, p2, r, r1, p
1     format ('   Range to which tank can see',/,
     1 ' Tank  HD      FE-S    FE-M    ranu')
2     format (i5,3f8.1,f8.4)
c
      if (trace) print *,'>detrg'
      if (keyd(1).ge.2) print 1
c     Find first and last firers on this side (narmy).
        IF (narmy.eq.BLU) THEN
          first = 1
          last = nblu
        ELSE
          first = nblu+1
          last = nblu+nred
        ENDIF
c     Loop thru all tanks on the side
        DO 80 tank = first,last
          p = ranu(0.0)
          DO 70 cond=1,3
            p1 = 1.0
c           Search for P-infinity values bounding x
              DO 60 krg=1,8
                p2 = pinfin(narmy,cond,krg)
                IF (p2 .lt. p) GOTO 65
                p1 = p2
60            CONTINUE
              p2 = 0.0
65            CONTINUE
c             Interpolate on p-infinity to find range.
                r1 = irginc*(krg-1)
                r  = r1 + irginc*(p1-p)/(p1-p2)
                rgvis(cond,tank) = r
70        CONTINUE
          if (keyd(1).ge.2) print 2,tank,(rgvis(cond,tank),cond=1,3),p
80      CONTINUE
      if (trace) print *,'<detrg'
      END
c V7.1
```

```
      FUNCTION DEVIC2 (attn, range)
c     Devic2: find resolvable cycles for device 2.
      real cofs(6), s(7)
      save cofs, foviw, s, tempd
      data tempd, foviw /2.5, .54/
      data cofs /.41089327, -.0892577, .026008138,
     1  -.004097143, .0003209958, -.00000982049/
      data s/4.798, .938, -.236, .011, .013, -.001, 0./
c
c     Calculate contrast
        y = -.45 + 1.19*alog10(attn)
        extcof = 10.**(y)
        amls = cofs(6)
        DO 30 i=5,1,-1
          amls = amls*range + cofs(i)
30      CONTINUE
        attn2 = extcof+amls
        cntrst = abs(tempd*exp(-attn2*range))
      IF (cntrst.gt.0.0112) THEN
c     Target/background is sufficient to detect
        clog = alog(cntrst)
c       $rc = sum from i=1 to 7 {s sub i clog sup {i-1}}$
          rc = s(7)
          DO 40 i=6,1,-1
            rc = rc*clog + s(i)
40        CONTINUE
        devic2 = amin1(rc/3.44,.9/foviw)
      ENDIF
      END
c V7.1
```

```
      SUBROUTINE DIS ENG (t, firer, tgt,drop,take)
c 7   Diseng: attempt to disengage 1 firer from 1 target.
c     Diseng is called by  impact if firer condition warrants.
c     When I include guns, other routines may call it.
      include 'common.h'
      integer armyf, armyt, tgt, firer
      logical in brst, hav amo, on tgt, drop, take, cango
3     format (f8.2,1x,a4,i3,' dis-engs ',a4,i3,20x,'#tgts=',i2)
c
      if (trace) print *,'>diseng'
c     Set useful local variables
        my tgt = nrtgt(firer)
        armyf = army(firer)
        armyt = 3-armyf
        hav amo = nrd(firer).lt.nrds(armyf)
        inbrst = nrpb(armyf).gt.1 .and. (0.ne.mod(nrib(firer),
     1    nrpb(armyf)))
        if (tgt.eq.FLS TGT) on tgt = .true.
        if (tgt.ne.FLS TGT) on tgt = fot(firer,tgt) .or.
     1     (kindrd(armyf).eq.4 .and. mot(firer,tgt))
      IF (on tgt) THEN
c     Firer on this target
        kind = kindrd(armyf)
        IF (kind.le.2 .or. kind.eq.5) THEN
          IF (nrpb(armyf).le.1) THEN
c         Single shot gun system or STAFF fire & forget system.
            IF (tgt.ne.FLS TGT) THEN
              if (fot(firer,tgt)) call cancel (firer,'fire  ',tgt)
              fot(firer,tgt) = .false.
            ENDIF
            hav amo = nrd(firer).lt.nrds(armyf)
            IF (hav amo) THEN
              thuman = 0.*exp(rolln(0.5))
              call selecs(t,firer,thuman)
            ELSEIF (can go(firer,t).and.ishtfs(armyf).gt.0) THEN
c           Firer moves on.
              if(keyd(1).ge.2)print 3, t,color(armyf),firer,
     1          color(armyt),tgt,nchan(firer)
              call skedul(t,firer,'accel ',NULL)
            ENDIF
            nrot(firer) = 0
            nrtgt(firer) = 0
          ELSE
c         Burst fire gun system.
            print *,'DISENG: Not implemented for burst fire guns.'
            STOP
          ENDIF
        ELSEIF(kind.eq.4) THEN
c       Guided missile system.
          if (drop) nchan(firer) = nchan(firer)-1
          IF (tgt.ne.FLS TGT) THEN
            IF (fot(firer,tgt)) THEN
              call cancel(firer,'fire  ',tgt)
              fot(firer,tgt) = .false.
            ENDIF
          ENDIF
          if(keyd(1).ge.2)print 3, t,color(armyf),firar,
     1      color(armyt),tgt,nchan(firer)
c         Firer attempts to select a new target
            IF (take) THEN
              call frdmsl(t,firer,tgt,armyf)
c             The firer begins to select a new target right now and
c             finishes the selection in a few seconds.
            ENDIF
        ENDIF
      ENDIF
      IF (.not.repeat) THEN
        repeat = .true.
        call skedul (t+.01,0,'search',NULL)
      ENDIF
      if (trace) print *,'<diseng'
      END
c V7.1
```

```
      FUNCTION DOT (a, b)
c 9   Dot: find dot product  a dot b.
      dimension a(3), b(3)
c
      dot = a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      END
c V7.3
```

```
      SUBROUTINE ENGAGE (t1, t2, firer, tgt)
c ?   Engage: begin engagement of a new tgt by this firer.
      include 'common.h'
      integer armyf, armyt, firer, tgt
1     format(' ENGAGE: armyf,ishtfs,firer,motion,STATNY',8i3)
c
      if (trace) print *,'>engage'
      armyf = army(firer)
      armyt = 3-armyf
      IF (life(firer).lt.FKILL.AND.nrd(firer).lt.nrds(armyf)  )THEN
       if(keym(18).gt.1)print 1,
     1 armyf,ishtfs(armyf),firer,motion(firer),STATNY
       nbrst(firer) = 1
       IF (ishtfs(armyf).gt.0 .AND. motion(firer).ne.STATNY
     1  .AND.  speed(armyf).gt.0.0) THEN
c       halt to fire
        call cancel (firer,'maxvel',NULL)
        call cancel (firer,'accel ',NULL)
        call skedul(t1,firer,'slowup',NULL)
      ELSE
c      Schedule a fire event otherwise
c       find range to target
         IF (tgt.eq.-1) THEN
          rg = rg0
CHANGED 1 Apr 86. Next line changed.
c         nrg = int((250.+rg)*.002)
          nrg = int(1.5+rg/irginc)
CHANGED 11 Jun 86 Preceding line changed to next line.
          nrg = int(0.5+rg/irginc)
         ELSE
          dm = rgf(t1,tgt,firer)
         ENDIF
        nrg = min0(8,nrg)
        dt = tfirst(army(firer),nrg) * exp(rolln(0.5))
        prev rd(firer) = 1
        nrib(firer) = 0
        nrot(firer) = 0
CHANGED 16 Jul 86  Next line added.
c       if(kindrd(armyf).eq.4) dt=0.1
change 23 Nov 89 by HLReed to make sure a round has been loaded
      t3 = amax1(tfire2(firer)+tmin(armyf),t2+dt)
      call skedul(t3,firer,'fire  ',tgt)
       ENDIF
      ENDIF
      IF (trace) print *,'<engage'
      END
c V7.1
```

```
      SUBROUTINE EVENTS
c 9   Events: call each event in sequence.
      include 'common.h'
      character*6 iwhat
1     format (' EVENTS: No such event type. Event="',a6,'"',
     1 ' Who=',i2,' Whom=',i2,' Time=',f7.2)
c
      if (trace) print *,'>events'
c     Initialize for battle
        call reset(keyd(5).gt.0)
        call creset
        call init
        tm lst = 0.0
c     Perform all events in the battle
10    CONTINUE
        call nextev (iwho, iwhat, iwhom, t)
        IF (iwhat.eq.'search') THEN
          call search (t)
        ELSEIF (iwhat.eq.'vanish') THEN
          call vanish (t,iwho,iwhom)
        ELSEIF (iwhat.eq.'appear') THEN
          call appear (t,iwho,iwhom)
        ELSEIF (iwhat.eq.'detect') THEN
          call detect (t,iwho,iwhom)
        ELSEIF (iwhat.eq.'select') THEN
          call select (t,iwho)
c     Change by HLReed to allow transfer of targets
        ELSEIF (iwhat.eq.'xfer  ') THEN
          call xfer(t,iwho,iwhom)
        ELSEIF (iwhat.eq.'fire  ') THEN
          call fire   (t,iwho,iwhom)
        ELSEIF (iwhat.eq.'impact') THEN
          call impact (t,iwho)
        ELSEIF (iwhat.eq.'slowup') THEN
          call slowup (t,iwho)
        ELSEIF (iwhat.eq.'halt  ') THEN
          call halt   (t,iwho)
        ELSEIF (iwhat.eq.'accel ') THEN
          call accelf (t,iwho)
        ELSEIF (iwhat.eq.'maxvel') THEN
          call maxvel (t,iwho)
        ELSEIF (iwhat.eq.'ikill ') THEN
          call latekl (t,iwho,iwhom)
        ELSEIF (iwhat.eq.'hide  ') THEN
          call hide   (t,iwho)
        ELSEIF (iwhat.eq.'reload') THEN
          call reload (t,iwho)
        ELSEIF (iwhat.eq.'popdn ') THEN
          call pop dn (t,iwho)
        ELSEIF (iwhat.eq.'finish') THEN
          call finish (tm lst)
          GOTO 99
        ELSE
          print 1,iwhat, iwho, iwhom, t
          STOP
        ENDIF
        tm lst=t
      GOTO 10
99    if (trace) print *,'<events'
      END
c V7.1
```

```
      FUNCTION EYE (alumnc, attn, range, visrg)
c     Eye: find resolvable cycles for the human eye. (Device 1)
      real a(4,7)
      save a, acon
c     sunset o'cast  heavy o'cast   overcast day   clear day
      data a/
     1 1.2378091942, 1.7176916034, 1.9909928015, 2.0892716525,
     2 0.4694720809,  .4739084812,  .4484981232,  .2813866389,
     3  .0493317078, -.2102695514, -.4084256747,-1.0084578626,
     4 -.0601756751, -.4161055149, -.6856409935,-1.4323484287,
     5 -.0558327470, -.2696921300, -.4318233767, -.8450225947,
     6 -.0174190671, -.0756229822, -.1197712507, -.2235482536,
     7 -.0018530403, -.0077222394, -.0121729428, -.0218136690/
      data acon /.4/
c
c     Find sky-to-ground ratio
        sog = (visrg+1.0)/3.
        sog = amin1(3.,amax1(1.,sog))
      cntrst = acon/(1.0+sog*(exp(attn*range)-1.0))
      IF (cntrst.ge.0.02) THEN
c     Target/Background contrast is sufficient to detect
        i = min0(4,1+int(alog10(alumnc)))
        ack = 10.**i
        j = min0(4,i+1)
        clog = alog(cntrst)
        rlo = a(i,7)
        rhi = a(j,7)
        DO 20 k=6,1,-1
          rlo = rlo * clog + a(i,k)
          rhi = rhi * clog + a(j,k)
20      CONTINUE
c       Interpolate & compute cycles across target
          eye = rlo+(rhi-rlo)*(alumnc-ack/10.)/(ack*.9)
      ENDIF
      END
c V7.2
```

```
        SUBROUTINE FINISH (t)
c 3     Finish: update statistics at end of a single engagement.
        include 'common.h'
        integer balive, dalive, ralive, brds, rrds
        dimension stata(8)
1       format(i6,2(5i3),4i3,1x,2f5.1,i9)
2       format('   Rep       Status of Combatants       ',
     1  ' Rds Used   Used/Tank',/
     1  '        |----Blue----| |-----Red----|  ',
     1  'by System   Blue  Red    seed',/
     1  6x,2(1x,'AL MO FO MF  K'),2x,'1  2  3  4')
c
        if (trace) print *,'>finish'
c       Count surviving blues and rounds fired
          balive = 0
          brds = 0
          dalive = 0
c      Change made by H.L. Reed on March 31, 1989 to keep decoys out of
c      win ratios and exchange ratio.  See also deaths.f
          DO 10 i=1,(nblu-ndecoy(BLU))
            k = life(i)
            if (k.ge.5) k=k-1
            nstats(k,BLU) = nstats(k,BLU)+1
            if (life(i).lt.FKILL) balive = balive+1
            brds = brds+nrd(i)
10        CONTINUE
          DO 11 i = nblu - ndecoy(BLU) +1, nblu
            if (life(i).lt.FKILL) dalive = dalive+1
11        CONTINUE
       call finsh2
c       Count surviving reds and red rounds fired.
          ralive = 0
          rrds = 0
          DO 20 i=1,(nred-ndecoy(RED))
            j = i+nblu
            k = life(i+nblu)
            if (k.ge.5) k=k-1
            nstats(k,RED) = nstats(k,RED)+1
            if (life(j).lt.FKILL) ralive = ralive+1
            rrds = rrds+nrd(j)
20        CONTINUE
          mystat(1) = mystat(1) + nblu-balive - ndecoy(BLU)
          mystat(2) = mystat(2) + ndecoy(BLU) - dalive
          nmov = 0
          ndow = 0
          DO 23, i = nblu+1, nblu+nred - ndecoy(RED)
          if(life(i).ge.FKILL.and.inwatch(i)) ndow = ndow + 1
          if(life(i).ge.FKILL.and..not.inwatch(i)) nmov = nmov +1
23        CONTINUE
          mystat(3) = mystat(3) + nmov
          mystat(4) = mystat(4) + ndow
c
        DO 30 i=1,4
30      stata(i) = 0.0
        j = nred-nwatch(scene)-ndecoy(RED)-nmov
        if (balive.gt.0 .and. j.eq.0) stata(1)=1.
        if (balive.eq.0 .and. j.gt.0) stata(2)=1.
        if (balive.gt.0 .and. j.gt.0) stata(3)=1.
        if (balive.eq.0 .and. j.eq.0) stata(4)=1.
        stata(5) = (nblu-ndecoy(BLU))-balive
        stata(6) = (nred-ndecoy(RED))-ralive
        stata(7) = float(brds)/float(nblu)
        stata(8) = float(rrds)/float(nred)
        excha = 0.0
        if (stata(5).gt.0.0) excha = stata(6)/stata(5)
        DO 40 i=1,8
          statb(i) = statb(i)+stata(i)
40      CONTINUE
c       Update ammo consumption for F-alive Blue tanks if combat occurred.
        IF (brds+rrds .gt.0) THEN
          DO 50 i=1,nblu
            j=nrd(i)+1
50        CONTINUE
        ENDIF
c
        if (keyd(1).ge.2 .or.
     1    (krep.eq.1 .and. keyd(1).eq.1)) print 2
        if (keyd(1).gt.0) print 1, krep, nstats, (nrd(i),i=1,4),
     1    stata(7), stata(8), irandm
        if (trace) print *,'<finish'
        END
c V7.1
```

```
      SUBROUTINE FINSH2
c 9   Finsh2: update statistics at end of a single engagement.
      include 'common.h'
c
      if (trace) print *,'>finsh2'
      DO 90 i=1,nblu
c       select blues for further combat
        IF (life(i).eq.ALIVE .and. nwaves.gt.1) THEN
          nsurv = nsurv+1
          nused(nsurv) = nrd(i)
        ENDIF
        IF (nrd(i).gt.nrds(BLU)-5) THEN
c       count systems with no & low ammo
          if (nrd(i).lt.nrds(BLU)) loammo = loammo+1
          if (nrd(i).ge.nrds(BLU)) noammo = noammo+1
        ENDIF
90    CONTINUE
      if (trace) print *,'<finsh2'
      END
c V7.1
```

```
      SUBROUTINE FIRE (t,firer,tgt)
c 7   Fire: Simulate firing of a round & schedule effects.
      include 'common.h'
      integer bullet, armyf, armyt, firer, tgt
1     format(f8.2, 1x, a4, i3, ' fires at ', a4, i3)
2     format(f8.2, 1x, a4, i3, ' ran out of ammo.')
c
      if (trace) print *,'>fire'
      busy(firer)=.false.
      IF (life(firer).ge.FKILL) THEN
       print *,'FIRE: firer',firer,' is F-killed or worse.'
       STOP
      ELSEIF (tgt.eq.0) THEN
       print *,'FIRE: firer',firer,' has no target.'
       STOP
      ELSE
c      Find nrs for tgt, army of firer, army of tgt
        armyf = army(firer)
        armyt = 3-armyf
       if (keyd(1).ge.7) print 1,t,color(armyf),firer,
   1    color(armyt),tgt
c      Update last firing time for firer & for firer at this tgt
        if (tgt.gt.0) tfire(firer,tgt) = t
        tfire2(firer) = t
c      Update positions, velocities and turret orientation
        IF (tgt.eq.-1) THEN
         rg = rg0
         nrg = max0(1,int(0.5+rg/irginc))
         s(1) = 0.0
         s(2) = rg0
         s(3) = 0.0
         if ((armyf.eq.BLU .and. scene.eq.BATTAK) .or.
   1      (armyf.eq.RED .and. scene.ne.BATTAK)) s(2) = -rg0
        ELSE
         dm = rgf(t,tgt,firer)
        ENDIF
        aspect(firer,TURRET) = anglef(VNORTH,s)
 c      Schedule any pinpoint detections
        call pinpnt (t,firer)
       IF (iflash(firer).eq.0) THEN
c      Branch for real firer (do nothing if firer is flashing decoy)
c       Create round with various attributes
        call create (10,bullet)
        a(bullet+1) = tgt
        a(bullet+2) = firer
        tfly = tof(armyf,nrg)
        t2 = t+tfly
        a(bullet+3) = s(1)+tfly*vt(1)
        a(bullet+4) = s(2)+tfly*vt(2)
        a(bullet+7) = psense(armyf,nrg)
        a(bullet+9) = vabs(vf)
        if (tgt.eq.-1) a(bullet+10) = 0.0
        if (tgt.gt.0) a(bullet+10) = vabs(vt)
        a(bullet+10)= vabs(vt)
        kshot(armyf,1) = kshot(armyf,1) + 1
c       Schedule impact for bullet
         call skedul (t+tfly,bullet,'impact',tgt)
         IF (kind rd(armyf).eq.4) THEN
          if (tgt.gt.0) mot(firer,tgt) = .true.
         DO 20 i=1,5
          IF (msl fly(armyf,firer,i) .eq. 0) GOTO 25
20       CONTINUE
         print *,'FIRE: Too many missiles'
         STOP
25       msl fly(armyf,firer,i) = bullet
       ENDIF
```

```
c         Update stowed rounds and expenditure
            nrd(firer) = nrd(firer)+1
            nrib(firer) = nrib(firer)+1
            if(nrib(firer).gt.nrpb(armyf)) nrib(firer)=1
            nrot(firer) = nrot(firer)+1
c         Move, fire, or switch targets as required
            IF (kind rd(armyf).le.2 .or. kind rd(armyf).eq.5) THEN
              IF (nrpb(armyf) .le.1) call frd ssg(t,firer,tgt,armyf)
            ELSEIF (kind rd(armyf) .eq. 4) THEN
c           Simultaneous missiles branch
              IF (nchan(firer).lt.nchans(armyf)) THEN
                call frd msl(t,firer,tgt,armyf)
c             ELSE
c               All guidance channels busy. Wait until impact.
              ENDIF
            ELSE
              print *,'FIRE: kind rd',kind rd(firer),' unknown.'
            ENDIF
          ENDIF
        ENDIF
         if (keyd(1).ge.2 .and. nrd(firer).ge.nrds(armyf)) print 2,
     1    t,color(armyf),firer
        if (trace) print *,'<fire'
        END
c V7.1
```

```
        SUBROUTINE FORCES
c 7     Forces: loop through desired blue/red force ratios.
        include 'common.h'
        integer range0
1       format(' SCENE#:',i3)
2       format(' Meeting engagement. #Blues =',i3,' #Reds =',i3)
3       format(' Red attack.         #Blues =',i3,' #Reds =',i3)
4       format(' Blue attack.        #Blues =',i3,' #Reds =',i3)
c
        if (trace) print *,'>forces'
        min blu = ntanks(scene,1)
        min red  = ntanks(scene,4)
        IF (min blu .gt. 0 .and. min red .gt. 0) THEN
          max blu = ntanks(scene,2)
          max red = ntanks(scene,5)
          inc blu = max0(1,ntanks(scene,3))
          inc red = max0(1,ntanks(scene,6))
          DO 50 nblu = min blu,max blu,inc blu
            DO 40 nred = min red,max red,inc red
              if (scene .eq. MEETNG) print 2, nblu, nred
              if (scene .eq. RATTAK) print 3, nblu, nred
              if (scene .eq. BATTAK) print 4, nblu, nred
              DO 30 range0 = min rg, max rg, inc rg
                rg0 = range0
                call waves (scene)
30            CONTINUE
40          CONTINUE
50        CONTINUE
        ENDIF
        if (trace) print *,'<forces'
        END
c V7.1
```

```
      SUBROUTINE FRD MSL (t, firer, tgt, armyf)
c 0   Frd msl: Fired a missile. now schedule effects.
      include 'common.h'
      logical done, tactc3, prflag
      integer armyf, firer, tgt
1     format('FRD MSL:  t,firer,tgt,armyf=',f7.2,3i3)
2     format(f8.2, 1x, a4, i3, ' begins to reload.')
3     format('FRD MSL: firer=',i3,' tactic=',i2,' #rds fired=',i2,
     1 ' #rds to fire=',i2)
c
      if (trace) print *,'>frd msl'
      prflag=.false.
      if (prflag) print 1,t,firer,tgt,armyf
      IF (nrd(firer).lt.nrds(armyf)) THEN
c     System has more rounds on board.
       if (prflag) print *, 'FRD MSL: nrd(firer)=',nrd(firer)
       IF (mod(nrd(firer),nipods(armyf)).gt.0 .or.
     1    nrd(firer).eq.0) THEN
c      System has more rounds in pod.
        if (prflag) print*,' FRD MSL: No reload. nrd, nipods=',
     1    nrd(firer),nipods(armyf)
        tactc3 = tactic(armyf).eq.3
        done = nrot(firer).eq.nrpt(armyf)
        if (prflag) print 3, firer, tactic(armyf),
     1    nrot(firer), nrpt(armyf)
        IF (tactc3 .and. done) THEN
c       Switch targets after firing a fixed nr of rds at it
         if (tgt.ne.FLS TGT) fot(firer,tgt) = .false.
         call selecs (t,firer,0.0)
        ELSE
c       Schedule next round fired
         timea = tmin(armyf)
         if (prflag) print *, 'FRD SSG: shoot again.'
         timeb = tfixed(armyf,nrg)
         timec = tmedin(armyf) * exp(rolln(0.5))
         dt = amax1(timea,timeb+timec)
         call skedul (t+dt,firer,'fire  ',tgt)
        ENDIF
       ELSE
c      Treat empty missile pod
        if (prflag) print *,' FRD MSL: Reloading'
        empty(firer) = .true.
        call cancel(firer,'fire  ',tgt)
        call cancel(firer,'select',NULL)
        nrot(firer) = 0
c       shud htf that is slowing to engage speed up now?
        call skedul (t+trelod(armyf),firer,'reload',NULL)
        if (keyd(1).ge.2) print 2,t,color(armyf),firer
       ENDIF
      ENDIF
      if (tgt.gt.0) fot(firer,tgt) = .false.
C     ABOVE LINE GOOD FOR MMSL THAT IS NOT LOADED W/ TARGETS
      if (trace) print *,'<frd msl'
      END
c V7 1
```

```
        SUBROUTINE FRD SSG (t, firer, tgt, armyf)
c 6     Frd ssg: Schedule effects after firing single shot gun.
        include 'common.h'
        logical can go, done, tactc3
        integer armyf, firer, tgt
1       format('FRD SSG:  t,firer,tgt,armyf=',f7.2,3i3)
2       format(f8.2, 1x, a4, i3, 'is out of ammo. Will attempt',
     1    ' to hide if mobile.')
c
        if (trace) print *,'>frd ssg'
        IF (nrd(firer).lt.nrds(armyf)) THEN
c       Have ammo branch
          tactc3 = tactic(armyf).eq.3
          done = nrot(firer).eq.nrpt(armyf)
          IF ((tactc3 .and. done)) THEN
c         Switch targets after firing a fixed nr of rds at it
            busy(firer) = .false.
            call dis eng (t, firer, tgt,.true.,.true.)
c           If no other tgt and can move, skedul acceleration
              if (can go(firer,t) .and. ishtfs(armyf).eq.1)
     1          call skedul(t,firer,'accel ',NULL)
            nrot(firer) = 0
          ELSEIF (tgt.gt.0) THEN
c         Schedule next round fired
            timea = tmin(armyf)
            timeb = tfixed(armyf,nrg)
            timec = tmedin(armyf) * exp(rolln(0.5))
            dt = amax1(timea,timeb+timec)
            call skedul (t+dt,firer,'fire  ',tgt)
          ENDIF
        ELSE
c       Out-of-ammo branch
          empty(firer) = .true.
          IF (cango(firer,t)) THEN
            call skedul (t,firer,'accel ',NULL)
            call skedul (t+thide(armyf),firer,'hide  ',NULL)
          ENDIF
        ENDIF
        if (trace) print *,'<frd ssg'
        END
c V7.2
```

```
      SUBROUTINE HALT (t, firer)
c 7   Halt: simulate tank halting.
      include 'common.h'
      logical cango, threat
      integer armyf, firer, tgt
1     format (f8.2,1x,a4,i3,' halts',12x,'(x=',f8.1,'   y=',f8.1',)')
2     format(' HALT:   firer, 0turrer, 0hull =', i3, 2f8.1)
3     format(' HALT: firer, tgt, armyf, nrg =', 4i3)
4     format(' HALT: rx1,rxe,tfirst,dt =',5f10.3)
5     format(' HALT: t, tlastx, dt =',5f10.3)
c
      if (trace) print *,'>halt'
      if(invisb.eq.1)call cancel (firer,'vanish',NULL)
      narmy = army(firer)
      call path (firer,t,motion(firer),0.0,x,y,vx,vy)
      if (keyd(1).ge.2) print 1, t, color(narmy), firer, x, y
      motion(firer) = STATNY
      tlastx = t-3.
      armyf = narmy
c     see if fire is a halt-tc-fire-system and can still shoot
      IF (ishtfs(armyf).eq.1  .and.
     1  life(firer).lt.FKILL .AND. nrd(firer).lt.nrds(armyf)) THEN
c       This is a halt-to-fire system. schedule firing if tgt
c       still available.
        threat = .false.
        IF (nrtgt(firer).eq.FLS TGT) THEN
          threat = knceal(firer).ne.FD
        ELSEIF (nrtgt(firer).gt.0) THEN
          threat = fot(firer,nrtgt(firer))
        ENDIF
        IF (.not.threat) THEN
c         firer's tgt has vanished. firer may move
          if(cango(firer,t))call skedul (t, firer, 'accel ', NULL)
        ELSE
          if (keyd(1).ge.2) print *,'HALT: tlastx, aspect needs wk!'
          rx1=rolln(0.5)
          rx2=exp(rx1)
          tgt=nrtgt(firer)
          dummy = rgf(t,firer,tgt)
          dt = tfirst(armyf,nrg)*rx2
c change Dec 89 by HLReed
          dt = amax1(dt,tfire2(firer)+tmin(armyf) -t)
          prev rd(firer) = 1
          nrib(firer) = 0
          nrot(firer) = 0
          call skedul (t+dt, firer, 'fire  ', tgt)
        ENDIF
      ENDIF
      if (trace) print *,'<halt'
      END
c V7.1
```

```
      SUBROUTINE HIDE (t, tgt)
c 5   Hide: Simulate tank hiding.
      include 'common.h'
      integer firer, tgt
1     format (f8.2,x,a4,i3,' goes into full defilade.')
c
      if (trace) print *,'>hide  '
      if (keyd(1).gt.1) print 1, t, color(army(tgt)), tgt
      knceal(tgt) = FD
c     Cancel all activities involving this tgt
c       except discard rounds-in-flight in the impact routine
        firer = 1
        if (tgt.le.nblu) firer=nblu+1
        last = nblu
        if (tgt.le.nblu) last=nblu+nred
        DO 20 i=firer,last
          los(i,tgt) = .false.
          los(tgt,i) = .false.
20      CONTINUE
        call newtgt (t, firer, tgt)
        call cancel (tgt,'all   ',NULL)
        call skedul(t,tgt,'slowup',NULL)
      call deaths(t)
      if (trace) print *,'<hide  '
      END
c V7.3
```

```
      SUBROUTINE IMPACT (t, bullet)
c 0   Impact: find what bullet does & what firer does.
      include 'common.h'
      logical loaded, hit
      integer bullet, expose
      atr(i) = a(bullet+i)

      if (trace) print *, '>impact'
c     Find useful variables.
        it = atr(1)
        I = atr(2)
        n = army(I)
        k = kindrd(n)
        expose = knceal(it)
        rgx = 0.0
c     Find what bullet does.
        IF (it.eq.FLS TGT) THEN
c       Round does nothing.
          kshot(n,4) = kshot(n,4)+1
        ELSEIF (expose.eq.FD .and. k.le.2) THEN
c       Count round hitting berm.
          kshot(n,5) = kshot(n,5)+1
          if (keyd(1).ge.2) print *, 'Tgt in full defilade.'
        ELSE
c       See if round hits.
          call mayhit(t,I,it,n,k,atr(9),atr(10),expose,hit)
        ENDIF
        a(bullet) = -a(bullet)
c     Find what firer does.
        IF (k.eq.4) THEN
c       Missile
c         Clear guidance channel.
            DO 20 j=1,5
              IF (mslfly(n,I,j).eq.bullet) GOTO 30
20          CONTINUE
            print *, 'IMPACT: Msl not assigned a channel.'
            print *, 'Channels assigned to',(mslfly(n,I,j),j=1,5)
            print *, 'Msl #=',bullet,' Contact Fred Bunn'
            STOP
30          CONTINUE
            mslfly(n,I,j) = NULL
          loaded = nchan(I).ge.nchans(n)
          call diseng (t,I,it,.true.,loaded)
          mot(I,it)=.false.
          fot(I,it)=.false.
          if (knceal(I).eq.HD .and. nchan(I).eq.0 .and. empty(I))
     1      call skedul(t,I,'popdn ',NULL)
        ELSE
c       KE, HEAT, or STAFF  [rethink this for STAFF]
          IF (it.eq.FLS TGT .or. hit.and.tactic(n).eq.2 .or.
     1      rgx.gt.4000.0) THEN
c           Switch targets if false target or rd hit & I switch on a hit.
c           Won't go here if I hit the berm; fls tgts don't go behind the
c           berm, and if true tgts do, the rd won't hit.
            ndet(I) = ndet(I)-1
            nrtgt(I)=0
            call diseng(t,I,it,.true.,.true.)
          ENDIF
        ENDIF
      if (trace) print *, '<impact'
      END
c V7.2
```

```
      INTEGER FUNCTION INDEXX(a, n, x)
c     --------------------------------
c     Find the index j, where a(j) <= x < a(j+1)
c     Adapted from Numerical Recipes, p90. The array ii must be increasing.
      integer n ,jl, ju, jm
      logical incres, above
      real a(n), x
c
      incres = a(n).gt.a(1)
      jl=0
      ju=n+1
10    IF (ju-jl.gt.1) THEN
        jm=(ju+jl)/2
        above=x.gt.a(jm)
        IF ((incres.and.above) .or. .not.(incres.or.above)) THEN
          jl=jm
        ELSE
          ju=jm
        ENDIF
      GOTO 10
      ENDIF
      indexx=jl
      END
c V7.1
```

```
      SUBROUTINE INIT
c 4   Init: Initialize scenario & schedule search at time zero.
      include 'common.h'
      integer firer, tgt
      logical regard
      common /cdetrg/ tdet(2)
      common /cregrd/ regard(NN)
c
      if (trace) print *,'>Init'
      call skedul(tmax,0,'finish',NULL)
      call deploy
      last = nred+nblu
      call init2 (1, nblu)
      call init2 (nblu+1, last)
      dm = rgf (0.0,1,1+nblu)
c     Set state variables for both red and blue systems.
      DO 30 firer=1,last
        busy(firer) = .false.
        empty(firer) = .false.
        serchg(firer) = .true.
        ndet(firer) = 0
        narmy = army(firer)
        ncol = knceal(firer)-1
        if(narmy.eq.BLU .and. scene.eq.BATTAK) ncol = 3
        if(narmy.eq.RED .and. scene.eq.RATTAK) ncol = 3
        cansee(firer) = rg.lt.rgvis(ncol,firer)
        regard(firer) = .true.
        DO 20 tgt=1,last
          foes(firer,tgt)= army(firer).ne.army(tgt)
          know(firer,tgt) = 0
          los(firer,tgt) = foes(firer,tgt) .and. invisb.ne.2
          mot(firer,tgt) = .false.
          fot(firer,tgt) = .false.
          seen(firer,tgt) = .false.
20      CONTINUE
30    CONTINUE
c     Hardwired values
        accmax(BLU) = 2.5
        accmax(RED) = 2.5
        wvlth(BLU)  = 50.
        wvlth(RED)  = 50.
      ampl(BLU) = accmax(BLU)/(TWOPI*speed(BLU)/wvlth(BLU))**2
      ampl(RED) = accmax(RED)/(TWOPI*speed(RED)/wvlth(RED))**2
      call serchl
      if (trace) print *,'<Init'
      END
c V7.1
```

```
      SUBROUTINE INIT2 (ifirst, last)
c 0   Init2: initialize each tank on one side.
      include 'common.h
      dimension iexpos(2,3)
      data iexpos /3, 3, 2, 3, 3, 2/
1     format(' INIT2: neval, nrd(1-3)=',4i5)
c
      if (trace) print *,'>init2'
      narmy = BLU
      if (ifirst.gt.1) narmy=RED
      last2 = nblu+nred
      jscene = iexpos(narmy, scene)
      DO 10 i=ifirst, last
        army(i) = narmy
        life(i) = ALIVE
        nrd(i) = 0
        nrtgt(i) = 0
        nchan(i) = 0
        nrot(i) = 0
        knceal(i) = jscene
c      Change introduced by HL Reed 8 Mar 89 to allow overwatch tanks to
c      be added to the attacking force.  See also subroutines deplo2,
c      input, and cango and common.h.
        if(inwatch(i)) knceal(i) = HD
        ichg(i) = 0
        motion(i) = MAXVL
        if(knceal(i).eq.HD .or. scene.eq.MEETNG) motion(i) = STATNY
c      End of 8 Mar 89 changes.
        nhot(i) = 0
        DO 8 j=1,5
          msl fly(narmy,i,j) = 0
          nstats(j,narmy) = 0
8       CONTINUE
        DO 10 j=1,last2
          tfire(i,j) = 0.0
c      Change by HLReed to make sure first round is loaded see ENGAGE also
          tfire2(i) = - tmin(narmy)
          kncels(i,j) = .false.
10      CONTINUE
      IF (ndecoy(narmy).gt.0) THEN
c     Change by H.L. Reed on 16 March 89 to correct decoys for Red Force
c     and to make flashing decoys usually be overwatching tanks.
        ldecoy = last - ndecoy(narmy) + 1
        DO 20 i = ldecoy, last
          iflash(i) = 0
          if (i.gt.last - nflash(narmy)) iflash(i)=1
          if(iflash(i).eq.0)nrd(i)=999
c     End of 16 March 89 change.
20      CONTINUE
      ENDIF
      call detrg(narmy)
      IF (invisb.eq.1) THEN
        if (narmy.eq.RED .and. scene.eq.RATTAK)
     1    call terain (ifirst,last)
        if (narmy.eq.BLU .and. scene.eq.BATTAK)
     1    call terain (ifirst,last)
      ELSE
        if (narmy.eq.BLU) call smoke
      ENDIF
c     Correct the nr of rounds used by blue systems
      IF (narmy.eq.BLU .and. nwaves.gt.1) THEN
        DO 40 i=1,nblu
          nrd(i) = nused(neval+i)
40      CONTINUE
c       print 1, neval,(nrd(i),i=1,3)
        neval = neval+nblu
      ENDIF
      if (trace) print *,'<init2'
      END
c V7.2
```

```
      SUBROUTINE INPUT
c 9   Input: read misc inputs
      include 'common.h'
      character*32 fname
      integer indx(5)
1     format(i1,a32)
4     format(a32)
c
c     Change introduced by H.L.Reed on 8 Mar 89 to allow overwatch
c     tanks to be added to the attacking force.  See also changes in
c     subroutines deplo2, init2, and cango and in common.h.
      read(5,*)(ntanks(1,j),j=1,6), nwatch(1)
      read(5,*)(ntanks(2,j),j=1,6), nwatch(2)
      read(5,*)(ntanks(3,j),j=1,6), nwatch(3)
c     End of 8 Mar 89 change
      read(5,*)(keyd(i),i=1,5)
      trace=keyd(4).gt.0
      read(5,*)indx
      DO 20 i=1,5
        if (indx(i).gt.1 .and. indx(i).le.20) keym(indx(i))=1
20    CONTINUE
      read(5,*)min rg, max rg, inc rg, irginc
      rgincr = irginc
      rginc2 = 0.5*irginc
      read(5,*) nreps, nwaves, iangd, meth sm, irandm
      read(5,*) tmax
      read(5,4) fname
      call rdmisc(fname,BLU)
c     Read pkh data for Blue. Change by HLReed for new vulnerability model
        read 1, ipkh, fname
        call rdpkh(fname,BLU)
      read(5,4) fname
      call rdmisc(fname,RED)
c     Read pkh data for Red.  Change by HLReed for new vulnerability model
        read 1, ipkh, fname
        call rdpkh(fname,RED)
      read(5,*) invisb,r
      IF (invisb.ne.1) THEN
        print *,' Smoke causes intervisibility.'
        read *
        read *, ((toutil(i,j),j=1,5),i=1,21)
        read *
        read *, ((toutvl(i,j),j=1,5),i=1,21)
        read *
        read *, ((touti(i,j),j=1,5),i=1,21)
        read *
        read *, ((toutv(i,j),j=1,5),i=1,21)
        read *
        read *, ((tini(i,j),j=1,5),i=1,21)
        read *
        read *, ((tinv(i,j),j=1,5),i=1,21)
      ENDIF
      if (trace) print *,'<input'
      END
c V7.1
```

```
      FUNCTION IZHIT (nbox, ndim, narmy, x, y,tgt, theta)
c 6   Iz hit: find if the target is hit.
      include 'common.h'
      integer tgt
1     format (' IZHIT: the round is high. y, ylimit, x =',3f7.3)
2     format (' IZHIT: the round is low.  y, ylimit, x =',3f7.3)
3     format (' IZHIT: the round is wide. y, ylimit =', 2f7.3,/
     1  '        x, xleft, xright = ', 3f7.3)
4     format (' IZHIT:  the round hits.    y, ylimit =', 2f7.3,/
     1  '        x, xleft, xright = ', 3f7.3)
c
      if (trace) print *,'>izhit'
      izhit = 0
      ylimit = sysdim(narmy,ndim)
      IF (ylimit.le.abs(y)) THEN
c       Too high or too low
        IF ( keym(6).gt.0)  THEN
          if (y.gt.0.0) print 1, y, ylimit, x
          if (y.le.0.0) print 2, y, ylimit, x
        ENDIF
      ELSE
c     Height ok
c       Find theta
          vtgt = vabs(vt)
          theta = aspect(tgt,nbox)*deg
c       Select tgt orientation between 0 & 360 deg.
        if(vtgt.gt.0 .and. nbox.eq.HULL)theta=anglef(VNORTH,vt)*deg
        phi = anglef(VNORTH,s)*deg
        theta1 = angsum(phi,-theta)
        theta = theta1/deg
        call bounds (narmy, nbox, theta, xleft, xright)
        if (x.gt.xleft .and. x.lt.xright) izhit = 1
        IF (keym(6).gt.0) THEN
            if(izhit.eq.0) print 3, y,ylimit,x,xleft,xright
            if(izhit.eq.1) print 4, y,ylimit,x,xleft,xright
        ENDIF
      ENDIF
      if (trace) print *,'<izhit'
      END
c V7.2
```

```
        SUBROUTINE KILL (firer, tgt, hit, injury,r)
c 9     Kill: find kill type for a hit on a tgt.
c       The routine is called by mayhit.
c       changed May 18,1989 for simplified hit and kill model, HL Reed
        include 'common.h'
        logical hit
        integer firer, tgt
        common /cpkh2/ pkill(2,3,2,5,9)
        save /cpkh2/
c   Change for interpolation on range for pkill
         p(i) = (1.0 - r) * pkill(narmy,ncase,nhdfe,i,jrg)
     1            + r * pkill(narmy,ncase,nhdfe,i,jrg+1)
        if (trace) print *,'>kill'
        nhdfe = knceal(tgt)-1
        narmy = army(firer)
        IF (motion(tgt) .ne. STATNY) THEN
             ncase = 3
        ELSE IF (motion(firer) .ne. STATNY) THEN
             ncase = 2
        ELSE
             ncase = 1
        END IF
c Find kill level
c   Change 12-9-89 by HLReed for interpolation on range for pkill
c   Get ratio based on 500 meter intervals
        r = r/500.
c   Is range > 4000 meters if so then use 3999.5
        if(r .GE. 8.) r = 7.999
c   Get integer part
        jrg = int(r)
c   and fractional part
        r = r - float(jrg)
c   Correct for the fact that indices start at 1 rather than 0
        jrg = jrg + 1
        temp = ranu(0.0)
        IF     (temp .gt. p(1)) THEN
c no hit and no kill
           hit    = .false.
           injury = ALIVE
        ELSEIF (temp .gt. p(2)) THEN
c a hit and a 'k' kill
           hit    = .true.
           injury = KKILL
        ELSEIF (temp .gt. p(3)) THEN
c a hit and an 'm&f' kill but no 'k'
           hit    = .true.
           injury = MFKILL
        ELSEIF (temp .gt. p(4)) THEN
c a hit and an 'f' kill but no 'm' kill
           hit    = .true.
           injury = FKILL
        ELSEIF (temp .gt. p(5)) THEN
c a hit and an 'm' kill but no 'f' kill
           hit    = .true.
           injury = MKILL
        ELSE
c a hit but no kill
           hit    = .true.
           injury = ALIVE
        ENDIF
          if (injury.eq.ALIVE) kshot(narmy,10) = kshot(narmy,10)+1
          if (injury.eq.MKILL) kshot(narmy,11) = kshot(narmy,11)+1
          if (injury.eq.FKILL) kshot(narmy,12) = kshot(narmy,12)+1
          if (injury.eq.MFKILL) kshot(narmy,13) = kshot(narmy,13)+1
          if (injury.eq.KKILL) kshot(narmy,14) = kshot(narmy,14)+1
        if (trace) print *,'<kill'
        END
c V7.2
```

```
      FUNCTION KILL5 (firer, tgt, x, y)
c 9   Kill5: find kill type for a STAFF-like round.
c     sector - angle band
c     band - range band (distance above tgt).
      include 'common.h'
      integer firer, tgt
      integer fan, fans, band, sector
      common /cpkh5/ anglim(4), pkh5(2,7,4,4,12), y1, y2, y3, fans
      save /cpkh5/
      dimension pksave(4)
1     format (' KILL5: narmy, ht,d,y1,y2,y3 =', i4,5f8.1)
2     format (' kill level=',i1,' ran=',f5.3,' p(m,f,mf,k)=',4f5.2)
3     format (' KILL5: band, anglex, sector, fan=',i4,f8.1,2i4)
4     format (' KILL5: x,y=',2f8.3)
c
      if (keyd(4).gt.0) print *,'>kill5'
      fan = nang
      if (nang.gt.fans) fan=14-nang
      pk = ranu(0.0)
      narmy = army(firer)
      ht = y+y2+y3
      d = sqrt(x**2 + ht**2)
      IF (d.le.y1+y2) THEN
         kill = ALIVE
      ELSEIF (d.ge.y1+y2+70.0) THEN
         kill = ALIVE
      ELSE
        band = 7-int((d-y1-y2)/10.0)
        anglex = atan2(abs(x),ht)*DEG
        anglex = 90.-anglex
        sector = indexx(anglex,4,anglim)
        pksave(1) =            pkh5(narmy,band,sector,1,fan)
        pksave(2) = pksave(1)+pkh5(narmy,band,sector,2,fan)
        pksave(3) = pksave(2)+pkh5(narmy,band,sector,3,fan)
        pksave(4) = pksave(3)+pkh5(narmy,band,sector,4,fan)
c     Find which kill type occurs.
        if (pk.lt.pksave(4)) kill = KKILL
        if (pk.lt.pksave(3)) kill = MFKILL
        if (pk.lt.pksave(2)) kill = FKILL
        if (pk.lt.pksave(1)) kill = MKILL
        if (pk.ge.pksave(4)) kill = ALIVE
        if (kill.eq.ALIVE) kshot(narmy,10) = kshot(narmy,10)+1
        if (kill.eq.MKILL) kshot(narmy,11) = kshot(narmy,11)+1
        if (kill.eq.FKILL) kshot(narmy,12) = kshot(narmy,12)+1
        if (kill.eq.MFKILL)kshot(narmy,13) = kshot(narmy,13)+1
        if (kill.eq.KKILL) kshot(narmy,14) = kshot(narmy,14)+1
      ENDIF
      kill5=kill
      if (keyd(4).gt.0) print *,'<kill5'
      END
c V7.1
```

```
      SUBROUTINE LATE KL (t, tgt,jj)
c 3   Late kl: Simulate recognition of a&f kill after period of inactivity.
      include 'common.h'
      integer firer, tgt
1     format(f8.2,1x,a4,i3,' I-killed.')
c
      if (trace) print *,'>latekl'
      if (keyd(1).gt.1) print 1, t, color(army(tgt)), tgt
      firer = 1
      if (tgt.le.nblu) firer=nblu+1
      life(tgt) = IKILL
      call cancel (tgt, 'ikill ',NULL)
        call newtgt (t,firer,tgt)
      call deaths(t)
      if (trace) print *,'<latekl'
      END
c V7.1
```

```
        SUBROUTINE MAX VEL(t, firer)
c 6     Max vel: simulate tank reaching cruise speed.
        include 'common.h'
        integer firer, tgt
1       format (f8.2,1x,a4,i3,' at full speed.')
c
        if (trace) print *,'>maxvel'
        if (keyd(1).ge.2) print 1, t, color(army(firer)), firer
        call path(firer,t,motion(firer),0.0,x,y,vx,vy)
        motion(firer) = MAXVL
        tgt = nrtgt(firer)
        IF (tgt.gt.0) THEN
          if (life(tgt).lt.IKILL) call engage(t,t,firer,nrtgt(firer))
        ENDIF
        if (trace) print *,'<maxvel'
        END
c V7.4
```

```
      SUBROUTINE MAYHIT (t,I,it,n,k,v1,v2,expose,hit)
c 0   Mayhit: Find what the round does.
c     Changed May 18, 1989 for simplified hit and kill model, HL Reed
      include 'common.h'
      logical hit
      integer expose

      if (trace) print *, '>mayhit'
      kshot(n,6) = kshot(n,6)+1
c     Find whether a hit occurs.
        hit = .false.
        rgx = rgf(t,I,it)
        r = rgx
        call kill(I,it,hit,injury,r)
      IF (hit) THEN
c     Treat hit.
        kshot(n,8) = kshot(n,8)+1
        if (life(it).eq.WFKILL) nhot(it)=nhot(it)+1
        if (nhot(it).gt.nbump(n)) call skedul(t,it,'ikill ',NULL)
        know(I,it)=2
        prevrd(I) = 2
        IF (reliab(n) .ge. ranu(0)) THEN
          call damage(t, I, it, injury)
        ELSE
c     Round is a dud.
          kshot(n,9) = kshot(n,9)+1
        ENDIF
      ELSE
c     Treat miss.
        kshot(n,7) = kshot(n,7)+1
        IF (psense(n,nrgf(rgx,rgincr)) .gt.ranu(0.0)) THEN
          prevrd(I) = 4
c    Changed by HLReed.  seems to be needed
        know(I,it)=1
          if (keyd(1).ge.2) print *,' Miss is sensed.'
        ELSE
          prevrd(I) = 3
          if (keyd(1).ge.2) print *,' Miss is not sensed.'
        ENDIF
      ENDIF
c     Careful. If either moving, make sure nx rd is treated as 1st
c       round if SS case occurs.
      if (vabs(vt).gt.0 .or. vabs(vf).gt.0) prevrd(I)=1
      if (v1.gt.0 .or. v2.gt.0) prevrd(I)=1
      if (trace) print *, '<mayhit'
      END
c Needs data checking.
c V7.1
```

```
      SUBROUTINE MK TBL (onlym,onlyf,fandm,mdisp,k,nrow)
c 3   Mk tbl: make head-on pkh table for echo.
      include 'common.h'
      common /cpkh/ table (4,12), echo(2,7,7), jrg(2,7), jdisp(2,7)
c
      if (trace) print *,'>mktbl'
      jrg(k,nrow) = (nrow-1)*irginc
      DO 11 j=1,4
        echo(k,nrow,j) = table(j,1)
11    CONTINUE
      echo(k,nrow,5) = onlym
      echo(k,nrow,6) = onlyf
      echo(k,nrow,7) = fandm
      jdisp(k,nrow) = mdisp
90    if (trace) print *,'<mktbl'
      END
c V7.2
```

```
        SUBROUTINE NEWTGT (t, firer, tgt)
c 3     New tgt: redirect all 'attackers' of tgt to a new target.
c       New tgt called for non-false tgts only and only if tgt condition
c       warrants it. It should only be called if tgt is V-killed,
c       vanishes, or hides.
c       Maybe it should be called if the tgt is I-killed by a gun system.
        include 'common.h'
        integer first, firer, tgt, armyf, armyt
        logical loaded, hav amo, cango
1       format(f8.2,1x,a4,i3,' dis-engs ',a4,i3,20x,'#tgts=',i2)
2       format(f8.2, 1x, a4, i3, ' begins to reload.')
c
        if (trace) print *,'>newtgt'
c       Find first and last 'attacker'
          first = 1
          if (firer.gt.nblu) first = nblu+1
          last = nblu
          if (firer.gt.nblu) last = nblu+nred
        armyf = army(first)
        armyt = 3-armyf
        kind = kindrd(armyf)
        nrpb2 = nrpb(armyf)
        DO 20 j=first, last
          IF ((mot(j,tgt) .or. fot(j,tgt)) .and. life(j).lt.FKILL) THEN
            IF (kind.le.2 .or. kind.eq.5) THEN
c           Single shot gun system or other fire & forget system.
              IF (nrpb(armyf).le.1) THEN
c             Single shot gun system.
                call cancel(j,'fire  ',tgt)
                if (nrtgt(j).eq.tgt) busy(j) = .false.
                if (nrtgt(j).eq.tgt) nrtgt(j) = 0
                hav amo = nrd(j).lt.nrds(armyf)
                IF (hav amo) THEN
                  thuman = 0.*exp(rolln(0.5))
                  call selecs(t,j,thuman)
                ELSEIF (can go(j,t).and.ishtfs(armyf).gt.0) THEN
c               Move out
                  call skedul(t,j,'accel ',NULL)
                ENDIF
                nrot(j) = 0
                fot(j,tgt) = .false.
                if (keyd(1).ge.2) print 1, t, color(armyf), j,
     1            color(armyt), tgt, nchan(j)
              ELSE
c             Burst fire gun.
                print *,'NEWTGT: Not implemented for burst fire.'
                STOP
              ENDIF
            ELSEIF (kind.eq.4) THEN
c           Guided missile branch.
              if (fot(j,tgt)) call cancel(j,'fire  ',tgt)
              if (mot(j,tgt)) call abort(t,j,tgt)
              loaded = nchan(j) .eq. nchans(armyf)
              IF ((.not.empty(j) .and. mot(j,tgt) .and. loaded) .or.
     1          (.not.empty(j) .and. fot(j,tgt))) THEN
                IF ((mod(nrd(j),nipods(armyf)) .gt. 0)  .or.
     1            fot(j,tgt)) THEN
c               More rds in pod
                  IF (fot(j,tgt)) THEN
                    call cancel(j,'select',NULL)
                    busy(j) = .false.
                    fot(j,tgt) = .false.
                  ENDIF
C                 if (tgt.ne.FLS TGT) fot(j,tgt) = .false.
                  thuman = 0.0*exp(rolln(0.5))
                  call selecs(t,j,thuman)
              ELSE
c             Treat empty missile pod
                empty(j) = .true.
                call cancel(j,'fire  ',tgt)
                call cancel(j,'select',NULL)
                busy(j) = .false.
                nrot(j) = 0
c               shud htf that is slowing to engage speed up now?
                call skedul (t+trelod(armyf),j,'reload',NULL)
              if (keyd(1).ge.2) print 2,t,color(armyf),j
              ENDIF
            ENDIF
            nchan(j) = nchan(j) -1
            nrtgt(j) = 0
            fot(j,tgt) = .false.
          ENDIF
        ENDIF
        if (seen(j,tgt)) ndet(j) = ndet(j) - 1
        seen(j,tgt) = .false.
20    CONTINUE
      IF (.not.repeat) THEN
        repeat = .true.
        call skedul (t+.01,0,'search',NULL)
      ENDIF
c     Following removed cause duplicates code in deaths routine
c     and is incorrect.
c     ndead=0
c     IF (armyt.eq.RED) THEN
c       DO 30 i=nblu+1,nblu+nred
c         if(life(i).gt.FKILL) ndead=ndead+1
c30     CONTINUE
c       if (ndead.eq.nred) call skedul(t+5.,NULL,'finish',NULL)
c     ELSE
c       DO 40 i=1,nblu
c         if(life(i).gt.FKILL) ndead=ndead+1
c40     CONTINUE
c       if (ndead.eq.nblu) call skedul(t+5.,NULL,'finish',NULL)
c     ENDIF
      if (trace) print *,'<newtgt'
      END
c V7.1
```

```
      SUBROUTINE NEXTEV (I,act,it,t)
c 0   Nextev: Find the next scheduled event.
      include 'clock.h'
      character*6 act
c
c     Fill arguments
        I = who(nxevnt)
        act = what(nxevnt)
        it = whom(nxevnt)
        t = when(nxevnt)
c     Drop storage unit from active storage chain
        n = nxevnt
        nxevnt = next(nxevnt)
c     Add storage unit to inactive storage.
        next(n) = nxidle
        nxidle = n
      END
c V7.1
```

```
      FUNCTION NRGF (rg,rgincr)
c 9   Nrgf: find which rgincr meter rg band range is in.
CHANGED 1 Apr 86. Next line changed.
c     nrgf = max0(1,int( (250.+rg)/500. ))
      nrgf = max0(1,int(0.5+rg/rgincr))
      END

c V7.2
```

```
      SUBROUTINE NXWAVE
c 9   Nx wave: Simulate all reps for the nth engagement.
      include 'common.h'
      character*4 str(3)
      dimension istatb(8)
      data str /'Wtg ','Ratk','Batk'/
c
      if (trace) print *,'>Nxwave'
      nrg = rg0/irginc
      nreps2 = nsurv/nblu
      neval = mod(nsurv,nblu)
      noammo = 0
      loammo = 0
      nsurv = neval
      DO 20 i=1,8
        statb(i) = 0.0
20    CONTINUE
c    Changed by HLReed for printing
      DO 21 i=1,4
        mystat(i) = 0
21    CONTINUE
      DO 30 nrep = 1,nreps2
        krep=nrep
        call events
30    CONTINUE
c     Update statistics after all reps of nth engagement.
c    Changed by HLReed for printing
      temp1 = float(mystat(1))/float(nreps2)
      temp2 = float(mystat(2))/float(nreps2)
      temp3 = float(mystat(3))/float(nreps2)
      temp4 = float(mystat(4))/float(nreps2)
        DO 40 i=1,8
          statc(i) = statc(i) + statb(i)
40      CONTINUE
        noamo2 = noamo2+noammo
        loamo2 = loamo2+loammo
        DO 50 i=1,4
          istatb(i) = 0.5 + 100*statb(i) / nreps2
          statb(i+4) = statb(i+4) / nreps2
50      CONTINUE
c    Changed by HLReed for printing
2     format(f5.0,f6.3,f6.3,f6.3,f6.3,i4,i4,f6.2,f8.3,f8.3)
          exchc = 0
          if (statc(5).gt.0) exchc = statc(6)/statc(5)
        print 2,rg0,temp1,temp2,temp3,temp4,istatb(1),istatb(2),
     1  exchc,statb(7), statb(8)
        nreps3 = nreps3+nreps2
      if (trace) print *,'<nxwave'
      END
c V7.2
```

```
      SUBROUTINE PATH (firer,t, motio2, delt, x, y, vx, vy)
c 4   Path: search path table for position and vel at time t.
      include 'common.h'
      logical is atkr, kan go, old
      integer firer
c
      if (trace) print *,'>path'
      narmy = army(firer)
      is atkr = (scene.eq.RATTAK .and. narmy.eq.RED) .or.
     1  (scene.eq.BATTAK .and. narmy.eq.BLU)
      kan go = (motio2.ne.STATNY .or.
     1  life(firer).eq.ALIVE .or. life(firer).eq.FKILL)
      dt = t-t0(firer)
      old = dt .gt. delt
      IF (is atkr .and. kan go .and. old) THEN
c     Update positions and velocity.
        t0(firer) = t
        if (motio2.eq.SLOWNG) THEN
          dv = decel(narmy)*dt
          y0(firer) = y0(firer)+dt*(vy0(firer)-0.5*dv)
          v = vy0(firer)-dv
          if (abs(v).lt.0.001) v = 0.0
          vy0(firer) = v
        ELSEIF (motio2.eq.STATNY) THEN
          vy0(firer) = 0.0
        ELSEIF (motio2.eq.ACCELG) THEN
          dv = accel(narmy)*dt
          y0(firer) = y0(firer)+dt*(vy0(firer)+0.5*dv)
          vy0(firer) = vy0(firer)+dv
        ELSEIF(motio2.eq.MAXVL) THEN
          y0(firer) = y0(firer)+vy0(firer)*dt
          vy0(firer) = speed(narmy)
        ELSE
          print *,'PATH: no such motion. motio2=,',motio2
          STOP
        ENDIF
        IF (accmax(narmy) .ne.0.) THEN
c       Add sinusoidal motion
          omega = TWOPI/wvlth(narmy)
          x0(firer) = ampl(narmy)*sin(omega*y0(firer))+
     1      xp(firer)
          vx0(firer)= ampl(narmy)*cos(omega*y0(firer))*
     1      omega*vy0(firer)
        ENDIF
      ENDIF
      x=x0(firer)
      y=y0(firer)
      vy=vy0(firer)
      vx=0.0
      if (trace) print *,'<path'
      END
c V7.2
```

```
      SUBROUTINE PINPNT (t,firer)
c 8   Pinpnt: Simulate firing signature (pinpoint) detection by some foes.
      include 'common.h'
      integer first, firer
      logical wilsee
1     format (f8.2,1x,a4,i3,' sees    ',a4,i3,' muzzle flash')
c
      if (trace) print *,'>pinpnt'
      first = 1
      if (firer.le.nblu) first = nblu+1
      last = nblu
      if (firer.le.nblu) last = nblu+nred
      pinpxx = pinp(army(first))
      DO 20 i=first, last
        wilsee = pinpxx.gt.ranu(0.0)
        IF (life(i).lt.FKILL .and. wilsee .and.
     2    ndet(i).lt.ndets(army(i)) .and.
     1    los(i,firer) .and. .not.seen(i,firer)) THEN
          if (keyd(1).ge.2) print 1,
     1      t, color(army(i)), i, color(army(firer)), firer
          seen(i,firer) = .true.
          ndet(i) = ndet(i) + 1
c       Change by HLReed added new variable for pinpoint time and
c       transfer of pinpoint detection
          thuman = pntime(army(i)) *exp(rolln(0.5))
          call selecs(t,i,thuman)
     if(xxfer(army(i))) call skedul(t+thuman,first,'xfer  ',firer)
        ENDIF
20    CONTINUE
      if (trace) print *,'<pinpnt'
      END
c V7.1
```

```
      SUBROUTINE POP DN (t,firer)
c 0   Pop dn: Have defender pop down to reload?
      include 'common.h'
      integer firer
c
      if (trace) print *,'>pop dn'
      call vanter(t,firer,NULL)
      if (trace) print *,'<pop dn'
      END
c V7.1
```

```
          SUBROUTINE PR GAME
c 9       pr game: print game control constants.
          include 'common.h'
1         format(21x,'#Blues     #Reds',/,
     1    ' Meeting engagement: ',3i2,4x,3i2,/,
     2    ' Red attack:         ',3i2,4x,3i2,/,
     3    ' Blue attack:        ',3i2,4x,3i2)
2         format (' DO rg=',3i5,'  (opening ranges)')
3         format (20i2)
4         format (' nreps =',i5,'  nwaves =',i3,'  iangd =',i2)
5         format (/,20('='),'RUN DESCRIPTION',20('='))
6         format (55('='),/)
c
          if (trace) print *,'>prgame'
          print 5
          print 1,((ntanks(i,j),j=1,6),i=1,3)
          print 2,minrg,maxrg,incrg
          print 3, keyd
          print 3, keym
          print 4, nreps, nwaves, iangd
          IF (meth sm.eq.1) THEN
             print *,'AMSAA MS error treatment.'
          ELSEIF (meth sm.eq.2) THEN
             print *,'BRL MS error treatment.'
          ELSE
             print *,'Error treatment value 'meth sm' wrong.'
          ENDIF
          print 6
          if (trace) print *,'<prgame'
          END
c V7.2
```

```
        INTEGER FUNCTION PRIORN (t, firer, lev old)
c 6     Priorn: select tgt with highest priority.
        include 'common.h'
        logical better, ck tgt
        logical pick
        integer firer, armyf
c
        if (trace) print *,'>priorn'
        armyf = army(firer)
c       'make' dummy tgt for comparison
          rg old=1.e35
          t old=1.e35
          lev old=1000
          priorn = NULL
        last = nblu+nred
        DO 30 mtgt=1,last
c       Compare all possible targets
          pick=.true.
c         Don't select this target if I'm already servicing it.
            if(mot(firer,mtgt).or fot(firer,mtgt))pick=.false.
          rg tgt = rgf (t,firer,mtgt)
          ck tgt = seen(firer,mtgt) .and. life(mtgt).lt.IKILL
     1      .and. rgtgt.le.4000.0 .and. pick
          IF (ck tgt) THEN
c         Firer sees tgt, it's threatening, & he's not firing at it.
            call priort(firer, mtgt, rg tgt, t, level)
c        Change by HLReed for way share is handled
           if(share(armyf)) then
           lll = 0
            do 20 jjj = 1,last
           if(mot(jjj,mtgt).or.fot(jjj,mtgt)) lll=30
20          continue
           level = level + lll
           endif
c           Now pick the tgt with highest priority
            rg tgt = rg tgt *(1+.05*rolln(1.0))
            t tgt = tfire(firer,mtgt)
            better = level .lt. lev old
            IF (lev old.eq.level) THEN
c           Same priority class; now break ties
c             if new tgts pick closer
              if (t tgt.le. 0) better = rg tgt .lt. rg old
c             if old tgts, pick older (least recently fired on)
              if (t tgt.gt. 0) better = t tgt .lt. t old
            ENDIF
         -  IF (better) THEN
              lev old = level
              t old = t tgt
              rg old = rg tgt
              priorn = mtgt
            ENDIF
          ENDIF
30      CONTINUE
        if (trace) print *,'<priorn'
        END
c V7.2
```

```
      SUBROUTINE PRIORT(firer, tgt, rg tgt, t, L)
c 0   PRIORT: find priority of tgt (w/ preference to old tgts)
      include 'common.h'
      integer firer, tgt
      dimension lev(21,2)
      save lev
      data lev/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
     *         1,2,3,4,5,6,7,8,9,15,16,17,18,19,20,21,10,11,12,13,14/
1     format(' PRIORT: ',a4,i3,' considrs ',a4,i3,' with priority',
     1  i4,'  (',i2,')')
c
      if (trace) print *,'>priort'
      j = nprior(army(firer))
      m = motion(tgt)
      t activ = 1.e35
      if (tfire2(tgt).gt.0.) t activ = t-tfire2(tgt)
      IF (tfire(firer,tgt).gt.0) THEN
c     firer has already shot at this target previously
        IF (know(firer,tgt).eq.1) THEN
c       Missed target with last round fired at it
          IF (rg tgt.lt.recknz(army(firer))) THEN
c         Target is within recognition range
            L = 5
            if (m.eq.STATNY .or. m.eq.SLOWNG) L = 3
            if (nrtgt(tgt).ne.0) L = 2
            if (t activ .lt. 30.) L = 1
          ELSE
c         Target is beyond recognition range
            L = 7
            if (m.eq.STATNY) L = 6
            if (t activ .lt. 30.) L = 4
          ENDIF
        ELSE
c       Hit target with last round fired at it
          IF (rg tgt.lt.recknz(army(firer))) THEN
c         Target is within recognition range
            L = 14
            if (m.eq.STATNY .or. m.eq.SLOWNG) L = 12
            if (nrtgt(tgt).ne.0) L = 11
            if (t activ .lt. 30.) L = 10
          ELSE
c         Target is beyond recognition range
            L = 16
            if (m.eq.STATNY) L = 15
            if (t activ .lt. 30.) L = 13
          ENDIF
        ENDIF
      ELSE
c     Target is a new target
        IF (rg tgt.lt.recknz(army(firer))) THEN
c       Target is within recognition range
          L = 19
          if (m.eq.STATNY .or. m.eq.SLOWNG) L = 18
          if (nrtgt(tgt).ne.0) L = 17
          if (t activ .lt. 30.) L = 8
        ELSE
c       Target is beyond recognition range
          L = 21
          if (m.eq.STATNY) L = 20
          if (t activ .lt. 30.) L = 9
        ENDIF
      ENDIF
      L = lev(L,j)
      if (trace) print *,'<priort'
      END
c V7.2
```

```
        SUBROUTINE PR MISC(narmy)
c 3     Pr misc: print misc tank characteristics.
        include 'common.h'
        integer irg(8)
10      format(/,' ====================BLUE SYSTEM DESCRIPTION',
     1  '====================')
11      format(/,' ====================RED SYSTEM DESCRIPTION',
     1  '====================')
12      format(/,
     1  ' SYSTEM DIMENSIONS',21x,'MOTION CHARACTERISTICS',/,
     1  ' Distance (m) from center of',
     1  11x,'Acceleration',f6.2,' m/s**2',/,
     1  ' turret ring to:',23x,'Deceleration',f6.2,' m/s**2',/,
     1  ' Turret top  ',f6.2,'  Ground    ',f6.2,
     1  '  Time to hide',f6.2,' sec',/,
     1  ' Turret Side ',f8.2,'  Hull side ',f8.2,/,
     1  ' Turret front',f8.2,'  Hull front',f8.2,/,
     1  ' Turret back ',f6.2,'  Hull back ',f6.2,/)
13      format(/,
     1  9x,'-------------DETECTION CAPABILITY------------',
     2  ' -----FIRING CYCLE----'
     1  ,/,'    Rg   Psense       P-detect             T-median   ',
     1  '    Tfirst  Tfixed   Tfly',/,
     1  '    (m)             HD    FE   FE-M      HD     FE   FE-M',
     1  '    (sec)    (sec)  (sec)',/,
     1  8(i7,2f7.2,2f6.2,f8.2,2f7.2,f7.1,f8.1,f8.2,/))
15      format(' System ammo load is',i3,' KE rounds')
16      format(' System ammo load is',i3,' HEAT rounds')
17      format(' System ammo load is',i3,' wait-til-impact missiles')
18      format(' System ammo load is',i3,
     1    ' simultaneous-flight missiles with',i2,'/pod &',/,
     2    ' reload time=',f7.1)
19      format(' System ammo load is',i3,' STAFF-like rounds')
20      format(' Systems can engage',i2,' targets simultaneously.')
21      format(
     1  ' Switch targets after:',/,' 1. A K-kill',
     2  ' (& don''t re-engage)',/,
     1  ' 2. An M&F-kill and',i2,' hits or',f5.1,' sec.',
     2  ' (& don''t re-engage)')
22      format(' 3. After scoring a hit.')
23      format(' 3. After',i3,' shots.')
24      format(' Return to partially serviced target after vainly',/,
     1  ' searching',f7.2,' sec for a new target.')
25      format(' System halts to fire.')
26      format(' System fires on the move.')

27      format(' Median time between rounds is',f6.2,' sec.')
28      format(' System fires',i2,' round bursts with rounds',
     1  ' spaced',f6.2,' sec apart.')
29      format(' There are', i3,' decoys,',i3,
     1  ' of which are flashing.')
30      format(' Minimum time to fire next round is',f6.2,' sec.')
31      format(' Probability of firing signature detection is',f6.2)
32      format(' Reliability is',f6.2)
33      format(' Recognizes targets inside',f8.0,' meters.')
34      format(' Probability of picking false HD, FE tgts are',2f6.3)
35      format(' Selects old, hit targets before new targets.')
36      format(' Selects new targets before old, hit targets.')
37      format(' Tgt sharing is set. Won''t pick a tgt being serviced.')
38      format(' Tgt sharing is off. Will pick a tgt being serviced.')
39      format(' Systems can detect',i2,' targets simultaneously.')
c
```

```
        if (trace) print *,'>prmisc'
        DO 50 i=1,8
          irg(i) = i*irginc
50      CONTINUE
c       Write header
          if (narmy.eq.1) print 10
          if (narmy.eq.2) print 11
c       Write system dimensions and motion characteristics
          print 12, accel(narmy),decel(narmy),
     1      sysdim(narmy,1), sysdim(narmy,5), thide(narmy),
     1      (sysdim(narmy,i),sysdim(narmy,i+4),i=2,4)
c       Write range dependent values
          print 13,
     1      (irg(i), psense(narmy,i), (pinfin(narmy,j,i),j=1,3),
     2      (tbar(narmy,j,i),j=1,3),
     1      tfirst(narmy,i),tfixed(narmy,i),
     1      tof(narmy,i),i=1,8)
c       Target switching policy
          i = tactic(narmy)
          print 21, nbump(narmy), tbump(narmy)
          if (i.eq.2) print 22
          if (i.eq.3) print 23, nrpt(narmy)
          if (i.gt.1) print 24, tlook(narmy)
          print 39, ndets(narmy)
          print 20, nchans(narmy)
          if (nprior(narmy).eq.1) print 35
          if (nprior(narmy).eq.2) print 36
          print *
c       Write projectile information
          if(kind rd(narmy).eq.1)print 15, nrds(narmy)
          if(kind rd(narmy).eq.2)print 16, nrds(narmy)
          if(kind rd(narmy).eq.3)print 17, nrds(narmy)
          if(kind rd(narmy).eq.4)print 18, nrds(narmy),
     1      nipods(narmy), trelod(narmy)
          if(kind rd(narmy).eq.5)print 19, nrds(narmy)
          if(ishtfs(narmy).gt.0)print 25
          if(ishtfs(narmy).eq.0)print 26
          print 27, tmedin(narmy)
          print 30, tmin(narmy)
          print 32, reliab(narmy)
          print 31, pinp(narmy)
          print 33, recknz(narmy)
          print 34, (pfalse(narmy,i),i=1,2)
          if(nrpb(narmy).gt.1)print 28, nrpb(narmy), rof(narmy)
          if(ndecoy(narmy).gt.0) print 29, ndecoy(narmy),
     1      nflash(narmy)
        if (share(narmy)) print 37
        if (.not.share(narmy)) print 38
        if (trace) print *,'<prmisc'
        END
c V7.2
```

```
      FUNCTION RANU (dm)
c     Ranu: A version of uran31 random uniform nr generator.
      common /crandm/ i, j
      real a1
      j=i
      j=j*25
      j=j-(j/67108864)*67108864
      j=j*25
      j=j-(j/67108864)*67108864
      j=j*5
      j=j-(j/67108864)*67108864
      a1=j
      i=j
      ranu= a1/67108864
      END
c V7.2
```

```
        SUBROUTINE RD MISC (dbname,narmy)
c 9     Rd misc: read miscellaneous tank characteristics.
        include 'common.h'
        character dbname*32, line*72
        real high(2)
2       format(1a1)
3       format(a)
c
        if (trace) print *,'>Rdmisc'
        open(4, file=dbname, status='old')
        rewind 4
        read(4,*) (sysdim(narmy,i),i=1,8)
        read(4,*) (psense(narmy,i),i=1,8)
c       Read nvl outputs.
          read(4,*) (pinfin(narmy,1,j),j=1,8)
          read(4,*) (pinfin(narmy,2,j),j=1,8)
          read(4,*) (pinfin(narmy,3,j),j=1,8)
          read(4,*) (tbar(narmy,1,j),j=1,8)
          read(4,*) (tbar(narmy,2,j),j=1,8)
          read(4,*) (tbar(narmy,3,j),j=1,8)
c    Change by HLReed  added pinpoint time input
        read(4,*) recknz(narmy),(pfalse(narmy,i),i=1,2),
     1    tlook(narmy),pinp(narmy),reliab(narmy),trelod(narmy),
     2    pntime(narmy)
        read(4,*)nrds(narmy),nrpt(narmy),nrpb(narmy),
     1    tactic(narmy),kind rd(narmy),nprior(narmy),
     2    nipods(narmy),nchans(narmy)
        read(4,*) (tof(narmy,i),i=1,8)
        read(4,*) (tfirst(narmy,i),i=1,8)
        read(4,*) tmedin(narmy), tmin(narmy), rof(narmy)
        read(4,*) (tfixed(narmy,i),i=1,8)
        read(4,*) accel(narmy), decel(narmy), speed(narmy),
     1    angle(narmy), thide(narmy)
        read(4,*) ishtfs(narmy), nbump(narmy), ibump
        tbump(narmy) = ibump
        read(4,*) ndecoy(narmy), nflash(narmy)
c    Change by HLReed  added xxfer for control of target transfer
        read(4,*) share(narmy), xxfer(narmy)
        read(4,2) kview(narmy)
        read(4,*) ndets(narmy)
        close (4)
        if (keyd(2).gt.0) call pr misc (narmy)
c       Convert tbar to detection probability / second.
          DO 30 i=1,8
            DO 20 j=1,3
              tbar(narmy,j,i) = 1.0-exp(-1.0/tbar(narmy,j,i))
20          CONTINUE
30        CONTINUE
        if (trace) print *,'<rdmisc'
        END
c V7.2
```

```
      SUBROUTINE RDPKH (dbname, narmy)
c 3   Rd pkh: read probability-of-kill data.
c     Changed for simplified hit and kill model May 19,1989, HL Reed
      include 'common.h'
      character*32 dbname
      common /cpkh2/ pkill(2,3,2,5,9)
      save /cpkh2/
      if (trace) write(*,*)'>rdpkh'
      open (4, file=dbname, status='old')
      rewind 4
      DO 100 ncase = 1,3
      DO 70 nhdfe=1,2
      DO 30 i=1,5
      read (4,*) n1,n2,n3,(pkill(narmy,ncase,nhdfe,i,j),j=1,9)
30    CONTINUE
70    CONTINUE
100   CONTINUE
      close(4)
90    if (trace) write(*,*)'<rdpkh'
      END
c V7.1
```

```
      SUBROUTINE RELOAD (t,firer)
c 6   Reload: simulates completion of reloading
c     30 Oct 85  Fixed statement printing error message
      include 'common.h'
      integer firer
      logical defndr
1     format(f8.2,1x,a4,i3,' finishes reloading')
2     format(f8.2,1x,a4,i3,' pops-up')
c
      if (trace) print *,'>reload'
      narmy = army(firer)
      if (keyd(1).ge.2) print 1,t,color(narmy),firer
      nrtgt(firer) = 0
      empty(firer) = .false.
      defndr = (scene.eq.BATTAK .and. narmy.eq.RED) .or.
     1  (scene.eq.RATTAK .and. narmy.eq.BLU)
      IF (defndr) THEN
c     Defender pops back up :nd will start searching.
        if (keyd(1).ge.2) print 2,t,color(narmy),firer
        call aprter(t,firer,tgt,HD)
      ELSE
c     Attacker or 'meeter' never popped down.
        thuman = 0.0*exp(rolln(0.5))
        call selecs(t,firer,thuman)
      ENDIF
      if (trace) print *,'<reload'
      END
c V7.1
```

```
        SUBROUTINE RESET (prflg)
c 0     Reset: Initialize the clock to time zero.
        include 'clock.h'
        logical prflg
c
        prflag = prflg
        nxevnt = 0
        nxidle = 1
        DO 10 j=1,NE
          next(j) = j+1
10      CONTINUE
        next(NE) = 0
        END
c V7.2
```

```
      FUNCTION RGF (t, firer, tgt)
c 3   Rgf: find the position of the firer w.r.t. the tgt.
      include 'common.h'
      integer firer, tgt
      common /pathc / xf, yf, xt, yt
      save /pathc /
1     format (9x,'Firer  x, y, vx, vy =', 4f10.1,/
     *        9x,'Target x, y, vx, vy =', 4f10.1)
c
      if (trace) print *,'>rgf'
      call path (firer,t,motion(firer),0.0,xf,yf,vf(1),vf(2))
      call path (tgt,t,motion(tgt),0.0,xt,yt,vt(1),vt(2))
      s(1) = xf-xt
      s(2) = yf-yt
      s(3) = 0.0
      vt(3) = 0.0
      vf(3) = 0.0
      temp = vabs(s)
      if(temp.GT.4000) temp = 4000
      nrg = nrgf(temp,rgincr)
      rgf = temp
      rg = irginc*nrg
      if (keym(20).gt.0) print 1,
     *   xf, yf, vf(1), vf(2), xt, yt, vt(1), vt(2)
      if (trace) print *,'<rgf'
      END
c V7.2
```

```
      FUNCTION RNDANG(iangd)
c     RNDANG: Draw a random angle from a cardioid/other distribution.
c
      PI=3.1415926536
      denom = 0.5/PI
      p=ranu(dummy)
c     Do binary search to find theta associated with random draw
        tlo =-PI
        if (iangd.gt.1) tlo = -PI/3.
        thi = PI
        if (iangd.gt.1) thi = PI/3.
        DO 20 i=1,10
          theta = 0.5*(tlo+thi)
          if (iangd.eq.1) px = (theta+sin(theta)+PI)*denom
          if (iangd.gt.1) px = (3.*theta+sin(3.*theta)+PI)*denom
          IF (px.lt.p) THEN
            tlo = theta
          ELSE
            thi = theta
          ENDIF
20      CONTINUE
      rndang=theta
      theta=theta*180./PI
      END
c V7.2
```

```
      FUNCTION ROLLN(sigma)
c 6   Rolln: find a random number from a normal distribution.
c     Box-Muller method
      save j, z
      data j/0/
c
      IF (j.eq.0) THEN
        x = sqrt(-2.*alog(ranu(dm)))
        y = 2.*3.1415926535*ranu(dm)
        rolln = x*cos(y)*sigma
        z = x*sin(y)
      ELSE
        j = 1-j
        rolln = z*sigma
      ENDIF
      END
c V7.3
```

```
      SUBROUTINE SEARC2 (t,firer,tgt,narmy,cond,dt)
c ?   Searc2: see if a tank detects a target during this second.
      include 'common.h'
      integer firer, tgt, cond
c
      if (trace) print *,'>searc2'
      temp = rg/rgincr
      indx = int(temp)
      IF (indx .lt. 1) THEN
        tlo = 1.0
        thi = tbar(narmy,cond,1)
      ELSEIF (indx .lt. 8) THEN
        tlo = tbar(narmy,cond,indx)
        thi = tbar(narmy,cond,indx+1)
      ELSE
        tlo = tbar(narmy,cond,8)
        thi = 0.0
      ENDIF
      frac = temp-aint(temp)
      pdetct = tlo + frac*(thi-tlo)
      IF (ranu(0.0).gt.pdetct) THEN
c     The firer doesn't detect the target in the next second.
        repeat = .true.
        dt = 1.0
      ELSE
c     This firer detects the target in this second.
        call skedul(t+ranu(0.0),firer,'detect', tgt)
      ENDIF
      if (trace) print *,'<searc2'
      END
c V7.3
```

```
        SUBROUTINE SEARCH (t)
c 3     Search: see if any targets are detected inl the next second.
        include 'common.h'
        logical ignore
        common /cserch/ i1,in,j1,jn,v,rgtbl(NN,NN),ignore(NN),rgvs(NN),
     1    ymax(NN),iarmy,jarmy,ndeti,ndetj,ni,nj
        save /cserch/
        rss(x,y) = sqrt(x*x+y*y)
c
        if (trace) print *,'>search'
        repeat = .false.
c       Update status of tanks.
c         (Next line shud eventually be updated in damage.f, ltkill.)
        DO 20 i=1,nblu+nred
          ignore(i) = ignore(i).or.life(i).ge.IKILL
          IF (.not.ignore(i) .and. motion(i).ne.STATNY) THEN
            call path(i,t,motion(i),0.0,dm,dm,dm,dm)
            DO 10 j=j1,jn
              rgtbl(i,j) = rss(x0(i)-x0(j),y0(i)-y0(j))
              rgtbl(j,i) = rgtbl(i,j)
10          CONTINUE
          ENDIF
20      CONTINUE
c
        DO 40 i=i1,in
c       Loop thru Southern tanks.
          IF (.not.ignore(i)) THEN
c         Consider tank i (It is alive and can detect or be detected.)
c  Change made March 20, 1989 by H.L.Reed to allow the individual condition
c  of each target tank to be used to define the probability of acquisition
            icond = 2
            if(motion(i).ne.STATNY) icond = 3
            if(knceal(i).eq.HD) icond = 1
            DO 30 j=j1,jn
              IF (.not.ignore(j)) THEN
c             Consider tank j (Also alive and can detect or be detected.)
                jcond = 2
                if(motion(j).ne.STATNY) jcond = 3
                if(knceal(j).eq.HD) jcond = 1
                rgi = rgvis(jcond,i)
                rgj = rgvis(icond,j)
                rgmax = amax1(rgi,rgj)
                rg = rgtbl(i,j)
                IF (rg.lt.rgmax) THEN
c               At least one is in detection rg of the other.
                  IF (los(i,j)) THEN
c                 Line-of-sight exists between them.
c                 Treat Southern tank as searcher
                    IF (rg.lt.rgi .and. .not.seen(i,j) .and.
     1                ndet(i).lt.ndeti) THEN
                      call searc2(t,i,j,iarmy,jcond,dt)
                    ELSE
                      repeat = .true.
                    ENDIF
```

```
c                 Treat Northern as searcher
                    IF (rg.lt.rgj .and. .not.seen(j,i) .and.
     1                ndet(j).lt.ndetj) THEN
                      call searc2(t,j,i,jarmy,icond,dt)
                    ELSE
                      repeat = .true.
                    ENDIF
                  ELSE
                    repeat = .true.
                  ENDIF
                ELSEIF (y0(i).gt.ymax(i)) THEN
c               Neither are in detection rg of the other.
                  call cancel(i,'all    ',ALL)
                  ignore(i) = .true.
                  if (keyd(1).ge.2) print *,' Cancel all events for',i
                  GOTO 40
                ENDIF
              ENDIF
30          CONTINUE
          ENDIF
40      CONTINUE
        if (repeat) call skedul(t+1.0,0,'search', NULL)
        if (trace) print *,'<search'
        END
c V7.1
```

```
      SUBROUTINE SELECS (t,firer,dt)
      include 'common.h'
      logical loaded
      integer firer, armyf
1     format (f8.2,1x,a4,i3,' does not select; selecting already.')
2     format (f8.2,1x,a4,i3,' does not select: channels full.')
3     format (f8.2,1x,a4,i3,' does not select; pod empty.')
4     format (f8.2,1x,a4,i3,' begins selection.')
c
      if (trace) print *,'>selecs'
      armyf = army(firer)
      if (kind.eq.4) loaded = nchan(firer).ge.nchans(armyf)
      if (kind.ne.4) loaded = nrtgt(firer).ne.0
      IF (busy(firer) .or. empty(firer) .or. loaded) THEN
c     Wait cause busy selecting, pod empty, or channels full.
        IF (keyd(1).ge.2) THEN
          IF (busy(firer)) THEN
            print 1, t, color(armyf), firer
          ELSEIF (loaded) THEN
            print 2, t, color(armyf), firer
          ELSEIF (empty(firer)) THEN
            print 3, t, color(armyf), firer
          ENDIF
        ENDIF
      ELSE
c     Start selection: none in progress and a channel is free.
        busy(firer) = .true.
        if (kind.eq.4) nchan(firer) = nchan(firer)+1
        call skedul(t+dt,firer,'select', NULL)
        if (keyd(1).ge.2) print 4, t, color(armyf), firer
      ENDIF
      if (trace) print *,'<selecs'
      END
c V7.2
```

```
          SUBROUTINE SELECT (t, firer)
c 6       Select: gunner chooses most dangerous target he sees.
          include 'common.h'
          character*4 colort
          logical tgt fls, f alive, can go
          integer firer, tgt, priorn, armyf
1         format(f8.2,1x,a4,i3,' selects  ',a4,i3,' with priority',i4,
     1      '  #tgts=',i2)
2         format(f8.2,1x,a4,i3,' selects  ',a4,' -1',
     1      ' & discards ',a4,i3, ' #tgts=',i2)
3         format(f8.2,1x,a4,i3,' selects',8x,'- (empty target set)')
4         format(' SELECT: ',a4,i3,' selects  ',a4,i3,' with priority',i4)
c
          if (trace) print *,'>select'
          armyf = army(firer)
          kind = kindrd(armyf)
          f alive = life(firer).lt.FKILL
          IF (f alive) THEN
c         Firer can shoot, so have him select.
            tgt = priorn(t,firer,level)
            IF (tgt.eq.NULL) THEN
c           Firer has no targets to select so he moves if possible
              if (keyd(1).ge.2) print 3, t,color(armyf), firer
              busy(firer) = .false.
              if (kind.eq.4) nchan(firer) = nchan(firer)-1
              IF (can go(firer,t) .and. (kind.le.2 .or.
     1          kind.eq.5 .or. nchan(firer).eq.0)) THEN
                call cancel(firer,'halt  ', NULL)
                call cancel(firer,'accel ', NULL)
                call skedul(t,firer,'accel ',NULL)
              ENDIF
            ELSE
c           Tgt has been selected
              colort = color(army(tgt))
              IF (tfire(firer,tgt).le.0.) THEN
c             Tgt is new; replace with false tgt randomly.
                i = knceal(tgt)-1
                pf = ranu(0)
                tgt fls = pf .lt. pfalse(armyf,i)
                IF (tgt fls) THEN
                  seen(firer,tgt) = .false.
                  if (keyd(1).ge.2) print 2, t, color(armyf),
     1              firer, colort, colort, tgt, nchan(firer)
                  tgt =  FLS TGT
c                 Restart search if it is turned off
                    IF (.not.repeat) THEN
                      repeat = .true.
                      call skedul(t,0,'search',NULL)
                   ENDIF
                ELSE
                  fot(firer,tgt) = .true.
                  if (keyd(1).ge.2) print 1, t, color(armyf),
     1              firer,colort,tgt,level,nchan(firer)
                ENDIF
              ELSE
c             Firer has previously serviced this target.
                fot(firer,tgt) = .true.
                if (keyd(1).ge.2) print 1, t, color(armyf),
     1          firer,colort,tgt,level,nchan(firer)
              ENDIF
              call engage (t, t, firer, tgt)
            ENDIF
            nrtgt(firer) = tgt
          ENDIF
          if (trace) print *,'<select'
          END
c V7.2
```

```
       SUBROUTINE SERCH0
c      Serch0: Find useful constants for search.
       include 'common.h'
       logical ignore
       common /cserch/ i1,in,j1,jn,v,rgtbl(NN,NN),ignore(NN),rgvs(NN),
     1 ymax(NN),iarmy,jarmy,ndeti,ndetj,ni,nj
       save /cserch/
       integer ncols(3)
       data ncols /2,3,1/

       if (trace) print *, '>serch0'
c      Find 1st and last in Southern & Northern forces.
       IF (scene.eq.BATTAK) THEN
         i1 = 1
         in = nblu
         j1 = nblu+1
         jn = nblu+nred
         v = speed(BLU)
       ELSE
         i1 = nblu+1
         in = nblu+nred
         j1 = 1
         jn = nblu
         v = 0.0
         if (scene.eq.RATTAK) v = speed(RED)
       ENDIF
c      Find actual detection ranges for targets.
         nb = ncols(scene)
         DO 20 i=1,nblu
           rgvs(i) = rgvis(nb,i)
           ignore(i) = .false.
20       CONTINUE
         nr = 4-nb
         DO 25 i=nblu+1,nblu+nred
           rgvs(i) = rgvis(nr,i)
           ignore(i) = .false.
25     CONTINUE
       iarmy = army(i1)
       jarmy = army(j1)
       ndeti = ndets(iarmy)
       ndetj = ndets(jarmy)
       ni = 2
       nj = 2
       if (scene.ne.MEETNG) ni = 3
       if (scene.ne.MEETNG) nj = 1
       if (trace) print *, '<serch0'
       END
c V7.2
```

```
      SUBROUTINE SERCH1
c     Find whether & when search should be started.
      include 'common.h'
      logical ignore
      common /cserch/ i1,in,j1,jn,v,rgtbl(NN,NN),ignore(NN),rgvs(NN),
     1 ymax(NN),iarmy,jarmy,ndeti,ndetj,ni,nj
      save /cserch/

      if (trace) print *, '>serch1'
      call serch0
      dtmin=1.0e10
c     Loop thru Southern force and Northern force.
      DO 40 i=i1,in
        ymin = 1.e10
        ymax(i) = -1.e10
        DO 30 j=j1,jn
          x = x0(j)-x0(i)
          y = y0(j)
          d = sqrt(x**2 + y**2)
          rgtbl(i,j) = d
          rgtbl(j,i) = d
          r = amax1(rgvs(i), rgvs(j))
          IF (r .gt. d) THEN
c         At least one is in detection range at time zero.
            ymin = amin1(ymin,0.0)
            ymax(i) = amax1(ymax(i),y+sqrt(r**2-x**2))
           ELSE
c         Neither is in detection range at time zero.
            IF (v .gt. 0.0 .and. abs(x) .lt. r) THEN
c           At least one will enter.
              q = sqrt(r**2 - x**2)
              ymin = amax1(0.0,amin1(ymin, y - q))
              ymax(i) = amax1(ymax(i), y + q)
            ENDIF
          ENDIF
30      CONTINUE
c       See if tank i should be ignored and update start time (dtmin).
          dt = 1.0e10
          if (ymin.eq.0.0) dt = 0.0
          if (v.ne.0.0) dt = ymin /v
          ignore(i) = dt.gt.tmax .or. ymax(i).le.0.0
          if (ignore(i)) call cancel(i,'all    ',ALL)
          IF (keyd(1).ge.2) THEN
            if (ignore(i)) print *,i,' Never enters detect rg of foe.'
            if (.not.ignore(i)) print *, i,
     1        ' Enters detect rg after traveling',ymin,' metres.'
          ENDIF
          dtmin = amin1(dtmin,dt)
40    CONTINUE
      if (dtmin.lt.tmax) call skedul(dtmin,ALL,'search',ALL)
      repeat = dtmin.lt.tmax
      if (trace) print *, '<serch1'
      END
c V7.1
```

```
      SUBROUTINE SKEDUL (t,I,act,it)
c 9   Schedule: Schedule an event for later execution.
      include 'clock.h'
      character*6 act
1     format(9x,'skedul ',i3,' ',a6,i3,' at time',f8.2)
c
      if (prflag) print 1, I, act, it, t
      IF (nxidle.eq.0) THEN
c     If storage all used stop
        print *,' Storage overloaded with too many events.'
        STOP
      ELSE
c     Store the event
c       Cut storage unit from empties
          n = nxidle
          nxidle = next(nxidle)
c       Then find where to insert this event in the event list.
        IF (nxevnt.le.0) THEN
c       New event is only event
          next(n) = 0
          nxevnt = n
        ELSE
c       Then find where to insert it.
c         Point to first 2 events
            l = nxevnt
            m = next(l)
c         Find where to insert them
            IF (t.ge.when(l)) THEN
c           See if between 2 scheduled events.
c             Loop till found.
20            IF (m.ne.0 .and. t.ge.when(m)) THEN
                l = m
                m = next(m)
                GOTO 20
              ELSE
c             Splice new event into list
                next(n) = m
                next(l) = n
              ENDIF
            ELSE
c           Place new event as most imminent
              next(n) = nxevnt
              nxevnt = n
            ENDIF
        ENDIF
c       Finally store event info
          when(n) = t
          what(n) = act
          who(n) = I
          whom(n) = it
      ENDIF
      END
c V7.1
```

```
      SUBROUTINE SLOW UP (t, firer)
c 8   Slow up: simulate tank starting to slow down.
      include 'common.h'
      integer firer
1     format (f8.2,1x,a4,i3,' continues to slow up.')
2     format (f8.2,1x,a4,i3,' would slow up if it weren''t',
    1   ' already stopped.')
3     format (f8.2,1x,a4,i3,' brakes',11x,'(was accelerating)')
4     format (f8.2,1x,a4,i3,' brakes',11x,'(was cruising)')
c
      if (trace) print *,'>slowup'
      kind mv = motion(firer)
      narmy = army(firer)
      IF (kind mv.eq.SLOWNG) THEN
c       Previous motion was slowing
        if(keyd(1).ge.2)print 1, t, color(narmy), firer
      ELSE IF (kind mv.eq.STATNY) THEN
c       Previous motion was stationary
        if(keyd(1).ge.2)print 2, t, color(narmy), firer
      ELSE IF (kind mv.eq.ACCELG) THEN
c       Previous motion was accelerating
        if(keyd(1).ge.2)print 3, t, color(narmy), firer
        call path (firer,t,motion(firer),0.0,x,y,vx,vy)
        dt = vy/decel(narmy)
        motion(firer) = SLOWNG
        call skedul(t+dt,firer,'halt  ', NULL)
      ELSE IF (kind mv.eq.MAXVL) THEN
c       Previous motion was cruising at max vel
        if(keyd(1).ge.2)print 4, t, color(narmy), firer
        call path (firer,t,motion(firer),0.0,x,y,vx,vy)
c       schedule halt time
        dt = vy/decel(narmy)
        call skedul(t+dt,firer,'halt  ', NULL)
        motion(firer) = SLOWNG
      ENDIF
      if (trace) print *,'<slowup'
      END
c V7.2
```

```
      SUBROUTINE SMOKE
c 0   Smoke: Find path lengths where attacker is hidden by smoke.
      include 'common.h'
      data ptbl /0.,.05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,
     1  .75,.8,.85,.9,.95,1.0/
      data rtbl /0.,1000.,2000.,3000.,4000./
c
      if (trace)print *,'>smoke'
      DO 80 nb=1,nblu
        DO 70 nr=nblu+1,nblu+nred
c       Find first time window for LOS between tanks nb, nr.
          p=ranu(dm)
          r=rg0
          if (kview(RED).eq.'V') dt=tdintp(ptbl,rtbl,toutv1,p,r,21,5)
          if (kview(RED).eq.'I') dt=tdintp(ptbl,rtbl,touti1,p,r,21,5)
          call skedul(dt,nb,'appear',nr)
          if (kview(BLU).eq.'V') dt=tdintp(ptbl,rtbl,toutv1,p,r,21,5)
          if (kview(BLU).eq.'I') dt=tdintp(ptbl,rtbl,touti1,p,r,21,5)
          call skedul(dt,nr,'appear',nb)
70      CONTINUE
80    CONTINUE
      if(trace)print *,'<smoke'
      END
c V7.1
```

```
      FUNCTION TDINTP(x1a, x2a, y, x1, x2, ix1a, ix2a )
c     TDINP:  Interpolates in a two dimensional matrix.
      integer ix1a, ix2a
      real y(ix1a,ix2a), x1a(ix1a), x2a(ix2a)
      integer j, k
      real y1, y2, y3, y4, t, u
c
      j = INDEXX( x1a, ix1a, x1 )
      k = INDEXX( x2a, ix2a, x2 )
      IF(k.eq.0) THEN
        PRINT*,'TDINTP: p,r,j,k=',x1,x2,j,k
        print *, x1a, x2a,ix1a,ix2a
      ENDIF
c
      y1 = y(j,k)
      y2 = y(j+1,k)
      y3 = y(j+1,k+1)
      y4 = y(j,k+1)
c
      t = (x1-x1a(j))/(x1a(j+1)-x1a(j))
      u = (x2-x2a(k))/(x2a(k+1)-x2a(k))
c
      TDINTP = (1-t)*(1-u)*y1 + t*(1-u)*y2 + t*u*y3 + (1-t)*u*y4
      END
c V7.2
```

```
      SUBROUTINE TERAIN (ifirst,last)
c 0   Mask st: find path lengths where attacker is masked by terrain
      include 'common.h'
      common /terane/ d(40), xold(20), yold(20), dist(20), iseg(20)
1     format (' visible for',f5.0,'m, then hidden for',f5.0,'m.')
c
      if (trace) print *,'>terain'
c     Find segment length at start of each engagement.
      DO 20 i=1,39,2
c       Hunfeld terrain constants
        f = -alog(ranu(0.0))
        d(i) = 300.*f**1.2
        f = -alog(ranu(0.0))
c       d(i+1) = 750.*f**2.0
        d(i+1) = 100.*f
        if (keyd(1).ge.2) print 1, d(i), d(i+1)
20    CONTINUE
c     Initialize data for each tank
      DO 30 i=ifirst,last
        call path (i,0.,motion(i),0.0,x,y,vx,vy)
        xold(i) = x
        yold(i) = y
        dist(i) = d(1)
        iseg(i) = 1
        call skedul (0.,i,'vanish',NULL)
30    CONTINUE
      if (trace) print *,'<terain'
c NOTES:
c 1. If meeting engagement skip this routine entirely
c 2. If side not moving skip at least part
      END
c V7.1
```

```
      FUNCTION VABS (a)
c 9   Vabs: find abslute value of a vector (magnitude).
      dimension a(3)
      vabs = sqrt(a(1)**2 + a(2)**2 + a(3)**2)
      END
c V7.3
```

```
      SUBROUTINE VANISH(t,tgt,firer)
c 0   Vanish: if tgt vanishes treat, otherwise reschedule vanish
      include 'common.h'
      integer tgt, firer
      common /terane/ d(40), xold(20), yold(20), dist(20), iseg(20)
      rss(x,y)=sqrt(x*x+y*y)
c
      if (trace) print *,'>vanish'
      narmy = army(tgt)
      IF (invisb.eq.1) THEN
        if(speed(narmy).le.0.)print *,'VANISH: narmy,speed=',narmy,
     1    speed(narmy)
        IF (speed(narmy).le.0.) STOP
        call path(tgt,t,motion(tgt),0.0,x,y,vx,vy)
c     Terrain causes intervisibility
        travel = rss(x-xold(tgt), y-yold(tgt))
        IF (travel.gt.dist(tgt)) THEN
c       Tgt is now masked by terrain
          xold(tgt) = x
          yold(tgt) = y
          iseg(tgt) = iseg(tgt)+1
          if (iseg(tgt).gt.40) iseg(tgt)=iseg(tgt)-40
          dist(tgt) = d(iseg(tgt))
          call vanter(t,tgt,firer)
          dt = dist(tgt)/speed(narmy) + 0.01
          call skedul (t+dt,tgt,'appear',NULL)
        ELSE IF (life(tgt).eq.ALIVE) THEN
c       Not yet masked by terrain, so reschedule
          dt = (dist(tgt) - travel) / speed(narmy) + 0.01
          call skedul (t+dt,tgt,'vanish',NULL)
        ENDIF
      ELSE
c     Tgt is now masked by smoke
          call vansmk(t,tgt,firer)
      ENDIF
      if (trace) print *,'<vanish'
      END
c V7.1
```

```
      SUBROUTINE VANSMK(t,tgt,firer)
c 0   Vansmk: Treat tgt that vanished behind smoke.
      include 'common.h'
      integer tgt, firer, armyf, armyt
1     format(f8.2,1x,a4,i3,' LOS to   ',a4,i3,' broken by smoke.')
c
      if (trace) print *,'>vansmk'
      armyt = army(tgt)
      armyf = 3-armyt
      if (keyd(1).ge.2) print 1, t, color(armyf), firer,
     1   color(armyt),tgt
c     Cancel line-of-sight between tgt and firer.
        los(firer,tgt) = .false.
        if (seen(firer,tgt)) ndet(firer) = ndet(firer)-1
        seen(firer,tgt) = .false.
        tfire(firer,tgt) = 0.0
CHANGED 2 Oct 86. Next line is new.
        if (busy(firer).and.nrtgt(firer).eq.tgt) busy(firer)=.false.
c     Abort firer missile on tgt.
        IF (mot(firer,tgt).or.fot(firer,tgt)) THEN
          call diseng(t,firer,tgt,.true.,.true.)
          if (mot(firer,tgt)) call abort(t,firer,tgt)
        ENDIF
c     Accelerate tgt that was halting to fire.
        IF (motion(tgt).eq.SLOWNG .and. life(tgt).eq.1 .and.
     1    fot(tgt,firer)) THEN
          call skedul (t,tgt,'accel ',NULL)
          call cancel (tgt,'halt  ',NULL)
        ENDIF
      if (trace) print *,'<vansmk'
c NOTE: shouldn't halted tgt accelerate too?
      END
c V7.3
```

```
        SUBROUTINE VANTER(t,tgt,firer)
c 0     Vanter: Treat tgt that vanished behind terrain.
        include 'common.h'
        integer tgt, firer
1       format(f8.2,1x,a4,i3,' vanishes',9x,'(x=',f8.1,'   y=',f8.1,')')
c
        if (trace) print *,'>vanter'
        narmy = army(tgt)
        if (keyd(1).ge.2) print 1, t, color(narmy), tgt,
     1    x0(tgt), y0(tgt)
        knceal(tgt) = FD
        nrtgt(tgt) = 0
CHANGED 22 Sep 86. Next line added.
        ndet(tgt) = 0
c       Cancel all lines-of-sight and sightings involving tgt
          DO 20 i=1,nblu+nred
            los(tgt,i) = .false.
            los(i,tgt) = .false.
            if (seen(i,tgt)) ndet(i)=ndet(i)-1
            seen(tgt,i) = .false.
            seen(i,tgt) = .false.
            tfire(tgt,i) = 0.0
            tfire(i,tgt) = 0.0
            fot(tgt,i) = .false.
c        Change by HLReed 1-12-90.  Seems to be needed to clear busy flag
c        GROUNDWARS has it also.
            busy(tgt) = .false.
20        CONTINUE
c       Abort outgoing missiles
          call abort(t,tgt,ALL)
          nchan(tgt) = 0
c       Abort incoming rounds & disengage tanks firing at tgt
          ifirst=1
          if (narmy.eq.1) ifirst = nblu+1
c         kind = kindrd(3-narmy)
          call newtgt(t,ifirst,tgt)
          call cancel (tgt,'fire  ',NULL)
CHANGED 27 Jun 85. Added next line.
          call cancel (tgt,'select',NULL)
c       Accelerate tgt that was halting to fire.
          IF (motion(tgt).eq.SLOWNG .and. life(tgt).eq.1) THEN
            call skedul (t,tgt,'accel ',NULL)
            call cancel (tgt,'halt  ',NULL)
          ENDIF
        if (trace) print *,'<vanter'
        END
c
c V7.1
```

```
        SUBROUTINE WAVES(n scene)
c 7     Waves: loop thru waves of red tanks.
        include 'common.h'
        dimension istatc(8)
2       format (8i10)
6       format(' Total',i4,i6,i5,4f5.1,2i3,17x,i9)
7       format(f6.2)
CHANGED 31 Mar 86. Next format added.
8       format(' Shots by:                Blue   Red',/,
     1  ' Fired                ',2i6,/,'  Wasted             ',2i6,/,
     2  '      Aborted         ',2i6,/,'     False tgts      ',2i6,/,
     3  '      Hidden tgts     ',2i6,/,'  Impacting          ',2i6,/,
     4  '      Misses          ',2i6,/,'     Hits            ',2i6,/,
     5  '         Duds         ',2i6,/,'        No damage    ',2i6,/,
     6  '         M-kill only  ',2i6,/,'        F-kill only  ',2i6,/,
     7  '         M&F-kill only',2i6,/,'        K-kill       ',2i6,/)
c
        if (trace) print *,'>Waves'
        IF (nreps*nblu.gt.3000 .and. nwaves.gt.1) THEN
          print *,'WAVES: Too many reps or blues.',nreps,nblu
        ELSE
c         Initialize scenario statistics
CHANGED 31 Mar 86. Next 4 lines added.
            DO 5 i=1,20
              kshot(1,i) = 0
              kshot(2,i) = 0
5           CONTINUE
            nwave = 0
            nreps3 = 0
            loamo2 = 0
            noamo2 = 0
            scene = nscene
            DO 22 i=1,8
              statc(i) = 0.
22          CONTINUE
CHANGED 1 Apr 86. Next line changed.
c           nrg = rg0/500
            nrg = rg0/irginc
            DO 25 i=1,3000
              nused(i) = 0
25          CONTINUE
            nsurv = nreps*nblu
c           call headrl
c         Loop thru up to 10 waves of red tanks
30          CONTINUE
              nwave = nwave+1
              call nxwave
            IF (nsurv.ge.nblu .and. nwave.lt.nwaves)    GOTO 30
c         Calculate summary statistics
            rpbs = statc(6)/(nreps*nblu)
            IF (nwave.gt.1) THEN
              DO 50 i=1,4
                istatc(i) = 0.5 + 100*statc(i)/nreps3
                statc(i+4) = statc(i+4)/nreps3
50            CONTINUE
              print 6, nreps3,
     1               (istatc(i),i=1,2),(statc(i),i=5,8),
     2         loamo2, noamo2, irandm
            ENDIF
CHANGED 31 Mar 86. Next line added
          kshot(1,2) =  kshot(1,3)+kshot(1,4)+kshot(1,5)
          kshot(2,2) =  kshot(2,3)+kshot(2,4)+kshot(2,5)
        ENDIF
        if (trace) print *,'<waves'
        END
```

```
      SUBROUTINE XFER(t,i1,j)
c     Change HLReed. Added Xfer for the transfer a detection of target j
c     to all vehicles on side starting with i1
      include 'common.h'
      i2 = nblu
      if (i1 .ne. 1) i2 = nblu+ nred
      DO 10 i = i1,i2
       IF (life(i) .lt. FKILL .and.
     1        ndet(i) .lt. ndets(army(i)) .and.
     2        los(i,j) .and.
     3        .not.seen(i,j) )   THEN
        seen(i,j) = .true.
        ndet(i) = ndet(i) + 1
        thuman = 0.0 + exp(rolln(0.5))
        call selecs(t,i,thuman)
       ENDIF
10    CONTINUE
      END
```

APPENDIX B

BASIC PROGRAM FOR VULNERABILITY TABLE

INTENTIONALLY LEFT BLANK.

Appendix B

BASIC program to produce vulnerability table from BRL vulnerability data and weapon accuracy.

```
100 DIM PKH(8,2,11,4,6), SIGMAX(3,3), SIGMAY(3,3), PK(3,2,4,8), CARD(6)
105 DIM PKILL(3,2,4,8)
106 DIM PH(8,3,2,6)      '(range,case,exposure,angle)
110 OPEN "v6808.doc" FOR INPUT AS #2  'typical name of file with BRL vul data
115 OPEN "v4_basea.doc" FOR OUTPUT AS #3
116 NSIG = 0     'nsig = 0 for base, nsig = 1,2,3 for var 1,2,3
120 INPUT #2, RR%, EX, DISP, TYPE, PKH(RR%/500,EX,DISP,TYPE,0)
130 FOR N = 1 TO 6
140 INPUT #2, PKH(RR%/500,EX,DISP,TYPE,N)
150 NEXT
160 INPUT #2, DUMMY
170 IF -(1+EOF(2)) GOTO 120
171 ' the following set of data are horizontal (sigmax) and vertical
172 ' (sigmay) dispersions in mils.  The first index is the variation
173 ' as picked by nsig, the second is the case where
175 ' case=1 is stat / stat; case=2 is moving firer; case=3 is moving tgt
180 SIGMAX(0,1) = .52   : SIGMAX(0,2) =2.59 : SIGMAX(0,3) =2.54
181 SIGMAX(1,1) = .41   : SIGMAX(1,2) = .63 : SIGMAX(1,3) =1.07
182 SIGMAX(2,1) = .40   : SIGMAX(2,2) = .45 : SIGMAX(2,3) = .88
183 SIGMAX(3,1) = .21   : SIGMAX(3,2) = .24 : SIGMAX(3,3) = .44
190 SIGMAY(0,1) = .52   : SIGMAY(0,2) =2.28 : SIGMAY(0,3) =1.57
191 SIGMAY(1,1) = .41   : SIGMAY(1,2) = .63 : SIGMAY(1,3) =1.07
192 SIGMAY(2,1) = .40   : SIGMAY(2,2) = .45 : SIGMAY(2,3) = .88
193 SIGMAY(3,1) = .21   : SIGMAY(3,2) = .24 : SIGMAY(3,3) = .44
195 VK = 1!    ' a factor that can be used to look at reduced lethality
200 HT = .375     ' half height of turret
210 WT1 = 1.175  ' half width of turret
220 LT1 = 1.475  ' half length of turret
230 HH = 1.5      ' height of hull
240 WH1 = 1.775   ' half width of hull
250 LH1 = 3.375   ' half length of hull
255 ' define the values of the cardioid distribution for 30 deg increments
260 CARD(0) = .1657
270 CARD(6) = .001
280 FOR N = 1 TO 5
290 THETA = .5236 * N
300 CARD(N) = .16667 + .16477 * COS(THETA)
310 NEXT
320 FOR CASE = 1 TO 3
330 FOR EX = 1 TO 2          ' hull defilade or fully exposed
340 FOR RANGE = 1 TO 8
350 SIGMAX = SIGMAX(NSIG,CASE) * RANGE * .5
351 SIGMAY = SIGMAY(NSIG,CASE) * RANGE * .5
360 D = 3.28*SQR(SIGMAX * SIGMAY)   ' dispersion in feet for vul table
361 IF D >= 11! THEN D = 10.999: ELSE IF D < 1! THEN D = 1!
365 D% = D - .5
370 IF D% > 10 THEN D% = 10
380 IF D% <  1 THEN D% =  1
385 D = D - D%
386 D1 = 1! - D
387 D1% = D% + 1
390 IF EX = 1 THEN GOSUB 630: ELSE GOSUB 720
400 FOR TYPE = 0 TO 4   ' type of kill where TYPE = 0 is Ph
410 TEMPPK = 0!
420 FOR N = 0 TO 6
430 IF TYPE = 0 THEN FACTOR = 1!:
       ELSE FACTOR = ( D1*PKH(RANGE,EX,D% ,TYPE,N)
                     + D *PKH(RANGE,EX,D1%,TYPE,N)) * VK
440 TEMPPK = TEMPPK + CARD(N) * PH(RANGE,CASE,EX,N) * FACTOR
450 NEXT   ' loop on N
460 PK(CASE, EX, TYPE, RANGE) = TEMPPK
470 NEXT   ' loop on TYPE
480 NEXT   ' loop on Range
490 NEXT   ' loop on EX
500 NEXT   ' loop on case
505 ' convert pk's to the intervals called p() in section 2.5.
510 FOR C% = 1 TO 3
```

```
520 FOR E% = 1 TO 2
530 FOR R% = 1 TO 8
535 PKILL(C%,E%,0,R%) = PK(C%,E%,0,R%)
540 PKILL(C%,E%,1,R%) = PK(C%,E%,0,R%)-PK(C%,E%,4,R%)
545 PKILL(C%,E%,2,R%) = PK(C%,E%,0,R%)-PK(C%,E%,1,R%)-PK(C%,E%,2,R%)+PK(C%,E%,3,R%)
550 PKILL(C%,E%,3,R%). = PK(C%,E%,0,R%)-PK(C%,E%,2,R%)
555 PKILL(C%,E%,4,R%) = PK(C%,E%,0,R%)-PK(C%,E%,3,R%)
570 NEXT            ' loop on R%
590 NEXT            ' loop on e%
595 NEXT            ' loop on c%
596 GOSUB 1510
600 END
610 '
620 ' Hull defilade hit probability given sigma x, sigma y, and n = theta/30
630 FOR N = 0 TO 6
640 THETA = .5236 * N
650 WT = WT1 * ABS(COS(THETA)) + LT1 * ABS(SIN(THETA))
660 GAUSSARG = HT/SIGMAY: GOSUB 860: PHTEMP = 2 * GAUSS -1
670 GAUSSARG = WT/SIGMAX: GOSUB 860: PH(RANGE,CASE,EX,N) = (2 * GAUSS -1)* PHTEMP
680 NEXT
690 RETURN
700 '
710 ' Exposed target hit probability given sigma x, sigma y, and n = theta/30
720 FOR N = 0 TO 6
730 THETA = .5236 * N
740 WT = WT1 * ABS(COS(THETA)) + LT1 * ABS(SIN(THETA))
750 GAUSSARG = (.3 + 2! * HT)/SIGMAY: GOSUB 860: PHTEMP = GAUSS
760 GAUSSARG = .3/SIGMAY: GOSUB 860: PHTEMP = PHTEMP - GAUSS
770 GAUSSARG = WT/SIGMAX: GOSUB 860: PH(RANGE,CASE,EX,N) = (2 * GAUSS -1)* PHTEMP ' ph turret
780 GAUSSARG = .3/SIGMAY: GOSUB 860: PHTEMP = GAUSS
790 GAUSSARG = (.3 - HH)/SIGMAY: GOSUB 860: PHTEMP = PHTEMP - GAUSS
800 WH = WH1 * ABS(COS(THETA)) + LH1 * ABS(SIN(THETA))
810 GAUSSARG = WH/SIGMAX: GOSUB 860: PH(RANGE,CASE,EX,N) = PHTEMP * (2 * GAUSS -1) + PH(RANGE,CASE,EX,N)
820 NEXT
830 RETURN
840 '
850 ' Normal Distribution Subroutine
860 TEMP = ABS(GAUSSARG)
870 GAUSS = .398942 * EXP(-.5 * GAUSSARG * GAUSSARG)
880 IF TEMP > 4.6844 THEN GOTO 940
890 TEMP = 1!/(1! + .2316419*TEMP)
900 GAUSS = 1! - GAUSS * TEMP * ((((1.33027 * TEMP - 1.821256) * TEMP
             + 1.781478) * TEMP - .3565638) * TEMP + .3193815)
910 IF GAUSSARG < 0 THEN GAUSS = 1! - GAUSS
920 RETURN
930 'Approximation for large values of the argument
940 GAUSS = 1! - GAUSS * (1!/TEMP - 1!/TEMP^3 + 3!/TEMP^5)
950 IF GAUSSARG < 0 THEN GAUSS = 1! - GAUSS
960 RETURN
1510 FOR C% = 1 TO 3
1520 FOR E% = 1 TO 2
1530 FOR T% = 0 TO 4
1540 PRINT #3, USING "# # # #.###";C%,E%,T%,PKILL(C%,E%,T%,1); ' range 0 is range 1
1550 FOR R% = 1 TO 8
1560 PRINT #3, USING " #.###";PKILL(C%,E%,T%,R%);
1570 NEXT           ' loop on R%
1575 PRINT #3, " "
1580 NEXT           ' loop on t%
1590 NEXT           ' loop on e%
1595 NEXT           ' loop on c%
1600 RETURN
```

No of
Copies Organization

1 Office of the Secretary of Defense
OUSD(A)
Director, Live Fire Testing
ATTN: James F. O'Bryon
Washington, DC 20301-3110

2 Administrator
Defense Technical Info Center
ATTN: DTIC-DDA
Cameron Station
Alexandria, VA 22304-6145

1 HQDA (SARD-TR)
WASH DC 20310-0001

1 Commander
US Army Materiel Command
ATTN: AMCDRA-ST
5001 Eisenhower Avenue
Alexandria, VA 22333-0001

1 Commander
US Army Laboratory Command
ATTN: AMSLC-DL
Adelphi, MD 20783-1145

2 Commander
US Army, ARDEC
ATTN: SMCAR-IMI-I
Picatinny Arsenal, NJ 07806-5000

2 Commander
US Army, ARDEC
ATTN: SMCAR-TDC
Picatinny Arsenal, NJ 07806-5000

1 Director
Benet Weapons Laboratory
US Army, ARDEC
ATTN: SMCAR-CCB-TL
Watervliet, NY 12189-4050

1 Commander
US Army Armament, Munitions
and Chemical Command
ATTN: SMCAR-ESP-L
Rock Island, IL 61299-5000

1 Commander
US Army Aviation Systems Command
ATTN: AMSAV-DACL
4300 Goodfellow Blvd.
St. Louis, MO 63120-1798

No of
Copies Organization

1 Director
US Army Aviation Research
and Technology Activity
ATTN: SAVRT-R (Library)
M/S 219-3
Ames Research Center
Moffett Field, CA 94035-1000

1 Commander
US Army Missile Command
ATTN: AMSMI-RD-CS-R (DOC)
Redstone Arsenal, AL 35898-5010

1 Commander
US Army Tank-Automotive Command
ATTN: AMSTA-TSL (Technical Library)
Warren, MI 48397-5000

1 Director
US Army TRADOC Analysis Command
ATTN: ATAA-SL
White Sands Missile Range, NM 88002-5502

(Class. only) 1 Commandant
US Army Infantry School
ATTN: ATSH-CD (Security Mgr.)
Fort Benning, GA 31905-5660

(Unclass. only) 1 Commandant
US Army Infantry School
ATTN: ATSH-CD-CSO-OR
Fort Benning, GA 31905-5660

1 Air Force Armament Laboratory
ATTN: AFATL/DLODL
Eglin AFB, FL 32542-5000

Aberdeen Proving Ground

2 Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, H. Cohen

1 Cdr, USATECOM
ATTN: AMSTE-TD

3 Cdr, CRDEC, AMCCOM
ATTN: SMCCR-RSP-A
SMCCR-MU
SMCCR-MSI

1 Dir, VLAMO
ATTN: AMSLC-VL-D

No. of
Copies Organization

2 Commander
US Army Laboratory Command
ATTN: AMSLC-TP-PB, I. Bartky.
F. Ostovic
Adelphi, MD 20783-1145

7 Commander
US Army, ARDEC
ATTN: SMCAR-TD
SMCAR-TDT
SMCAR-SCF, V. Galgano
E. Del Coco
M. Barbarisi
SMCAR-CCS
SMCAR-CCL-FW, H. Kahn
Picatinny Arsenal, NJ 07806-5000

1 Commander
US Army Watervliet Arsenal
ATTN: SMCWV-QAR, Building 44
W. Jarrett
Watervliet, NY 12189

6 Commander
US Army Tank Automotive Command
ATTN: AMSTA-RY, R. Beck
AMCPM-BFVS
AMCPM-ABMS
AMCPM-ABMS-SW
AMCPM-ABMS-SI,
G. Vander Waerden
Warren, MI 48090

2 Commandant
US Army Infantry School
ATTN: ATSH-TSM-FV
ATSH-CD-MLS-M
Fort Benning, GA 31095

No. of
Copies Organization

3 Commander
US Army Armor Center
ATTN: ATSB-CD
ATZK-CD-MS, Mr Falkovich
ATZK-AE-PD, Mr. Wells
Fort Knox, KY 40121

1 Commander
TRAC RPD
ATTN: ATRC-DCS
Fort Monroe, VA 23651

1 Director
Development Center
ATTN: MCDEC/DO92,
Firepower Division
Quantico, VA 22134

1 Director
Benet Weapons Laboratory
US Army, ARDEC
ATTN: SMCAR-CCB
Watervliet, NY 12189

2 PM-TMAS
ATTN: DRCPM-TMA-PA, K. Rubin
DRCPM-TMA, F. Steinberg
Picatinny Arsenal, NJ 07806

1 OUSD(A)
ATTN: Executive Director, DSB
Room 3D1020, Pentagon
Washington, DC 20301

1 OUSD(A)
ATTN: DUSD(R&AT)
Room 3E114, Pentagon
Washington, DC 20301

1 OUSD(A)
ATTN: DUSD(TWP)
Room 3E1044, Pentagon
Washington, DC 20301

No. of
Copies Organization

2 ADUSD
ATTN: TWP/LW, Room 3D1049/Viilu
TWP/AW, Room 3E1049/Loder
Pentagon
Washington, DC 20301

1 Assistant Secretary of Defense of C^{3I}
Room 3E172, Pentagon
Washington, DC 20301

1 OUSD(A)
ATTN: Director, Program Integration
Room 3E1034/Christie, Pentagon
Washington, DC 20301

1 Director, PA&E
ATTN: Director, LFD
Room 2B256, Pentagon
Washington, DC 20301

1 Director
DARPA
1400 Wilson Blvd.
Rosslyn, VA 22209

1 HQDA (SAUS-OR, Mr. Hollis)
WASH DC 20310-0001

1 HQDA (SARD-ZD)
WASH DC 20310-0001

1 Commander
USAMC
ATTN: AMCCG
5001 Eisenhower Avenue
Alexandria, VA 22333-0001

1 Commander
AVSCOM
ATTN: AMCPM-AAH
4300 Goodfellow Blvd.
St. Louis, MO 63120-1798

1 Commander
CECOM
ATTN: AMSEL-RD-EW-D
Fort Monmouth, NJ 07703-5303

No. of
Copies Organization

1 Commander
CNVEO
ATTN: AMSEL-RD-NV-T
Fort Belvoir, VA 22060-5166

1 Commander
ASL/LABCOM
ATTN: SLCAS-D
White Sands Missile Range, NM 88002-5501

1 Commander
VAL/LABCOM
ATTN: SLCVA-D
White Sands Missile Range, NM 88002-5513

2 Commander
US Army Missile Command
ATTN: AMCPEO-FS
AMSMI-RD-AS
Redstone Arsenal, AL 35898-5001

1 Director
Survivability Management Office
ATTN: SLCSM-GS
2800 Powder Mill Road
Adelphi, MD 20783-1145

1 Commander
USA FSTC
ATTN: William Rich
220 Seventh St., NE
Charlottesville, VA 22901-5396

1 Commander
TRAC RPD
ATTN: ATCG
Fort Monroe, VA 23651-5000

1 Commander
USA Air Defense Artillery School
ATTN: ATSA-CD
Fort Bliss, TX 79916

No. of
Copies Organization

1 Commander
USA Armor Center and Fort Knox
ATTN: ATSB-CD
Fort Knox, KY 40121-5000

1 Commander
USA Combined Arms Combat Development
Activity
ATTN: ATZL-CA
Fort Leavenworth, KS 66027-5300

1 Commander
USAAVNC and Fort Rucker
ATTN: ATZQ-TDS
Fort Rucker, AL 36362-5163

1 Commandant
USA Infantry School
ATTN: ATSH-CD-MLS
Fort Benning, GA 31905-5000

1 Commander
USA Intelligence Center and School
ATTN: ATSI-CD-TE
Fort Hauchuca, AZ 85613-7000

1 Director
TRAC-WSMR
White Sands Missile Range, NM 88002-5502

1 Assistant Secretary of the Navy
Research, Engineering and Systems
Room 4E736, Pentagon
Washington, DC 20350

1 Commandant
USMC
ATTN: Code RD
Washington, DC 20380-0001

1 Deputy Commanding General
MCRDAC
ATTN: Code D032
Quantico, VA 22134-5080

1 Director
USMC OTEA
Quantico, VA 22134

No. of
Copies Organization

1 Director
DIA
ATTN: RTS-2, Col MacNicoll
Pentagon
Washington, DC 20301-6111

2 Director
CIA
ATTN: Mr. John Ewen/Mr. Robert Gomez
P.O. Box 1925
Washington, DC 20505

1 Joint Electronic Warfare Center
ATTN: MAJ Harry Ladewig
San Antonio, TX 78243-5000

3 Los Alamos National Laboratory
ATTN: CDT (J. Immele)
ATAC (F. Day, P. Howe)
Los Alamos, NM 87545

1 Lawrence Livermore National Laboratory
ATTN: Dr. Milton Finger
Livermore, CA 94550

1 Geometric Solutions
ATTN: Mr. Harry L. Reed, Jr.
100 Custus Street
Aberdeen, MD 21001

Aberdeen Proving Ground

1 Dir, USAMSAA
ATTN: Mr. T. Ruth

1 Dir, USAHEL
ATTN: J. Waugh

1 Cdr, USACSTA
ATTN: STCS-CC-P, P. McCall

1 Commander
Project Manager Smoke/Obscurants
ATTN: AMCPEO-CNS-CT

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number BRL-CR-638 Date of Report AUGUST 1990

2. Date Report Received ____________________

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ____________________

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) ____________________

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. ____________________

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) ____________________

	Name
CURRENT ADDRESS	Organization
	Address
	City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

	Name
OLD ADDRESS	Organization
	Address
	City, State, Zip Code

(Remove this sheet, fold as indicated, staple or tape closed, and mail.)

--FOLD HERE--

DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066
OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT No 0001, APG, MD

POSTAGE WILL BE PAID BY ADDRESSEE

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989

--FOLD HERE--